Final Project Report

Prediction and Analysis of Liver Patient Data Using Machine Learning

1. Introduction

1.1. Project overview

In India, delayed diagnosis of diseases is a fundamental problem due to a shortage of medical professionals. A typical scenario, prevalent mostly in rural and somewhat in urban areas is:

1. A patient going to a doctor with certain symptoms.
2. The doctor recommending certain tests like blood test, urine test etc depending on the symptoms.
3. The patient taking the aforementioned tests in an analysis lab.
4. The patient taking the reports back to the reports back to the hospital, where they are examined and the disease is identified.

The aim of this project is to somewhat reduce the time delay caused due to the unnecessary back and forth shuttling between the hospital and the pathology lab. Historically, work has been done in identifying the onset of diseases like heart disease, Parkinson's from various features a machine learning algorithm will be trained to predict a liver disease in patients.

1.2. Objectives

The aim of this project is to somewhat reduce the time delay caused due to the unnecessary back and forth shuttling between the hospital and the pathology lab. A machine learning algorithm will be trained to predict a liver disease in patients.

2.Project Initialization and Planning Phase

The "Project Initialization and Planning Phase" marks the project's outset, defining goals, scope, and stakeholders. This crucial phase establishes project parameters, identifies key team members, allocates resources, and outlines a realistic timeline. It also involves risk assessment and mitigation planning. Successful initiation sets the foundation for a

well-organized and efficiently executed machine learning project, ensuring clarity, alignment, and proactive measures for potential challenges.

2.1. Define Problem Statement

The problem statement is formally defined as:

‘Given a dataset containing various attributes of 584 Indian patients, use the features available in the dataset and define a supervised classification algorithm which can identify whether a person is suffering from liver disease or not. This data set contains 416 liver patient records and 167 non- liver patient records.The data set was collected from north east of Andhra Pradesh, India. This data set contains 441 male patient records and 142 female patient records. Any patient whose age exceeded 89 is listed as being of age "90".

2.2. Project Proposal (Proposed Solution)

This seems to be a classic example of supervised learning. We have been provided with a fixed number of features for each data point, and our aim will be to train a variety of Supervised Learning algorithms on this data, so that , when a new data point arises, our best performing classifier can be used to categorize the data point as a positive example or negative. Exact details of the number and types of algorithms used for training is included in the 'Algorithms and Techniques' sub-section of the 'Analysis' part.

2.3. Initial Project Planning

In problems of disease classification like this one, simply comparing the accuracy, that is, the ratio of correct predictions to total predictions is not enough. This is because depending on the context like severity of disease, sometimes it is more important that an algorithm does not wrongly predict a disease as a non-disease, while predicting a healthy person as diseased will attract a comparatively less severe penalty.

3.Data Collection and Preprocessing Phase

The Data Collection and Preprocessing Phase involves executing a plan to gather relevant Liver Patient data from Kaggle, ensuring data quality through verification and addressing missing values. Preprocessing tasks include cleaning, encoding, and organizing the dataset for subsequent exploratory analysis and machine learning model development.

3.1. Data Collection Plan and Raw Data Sources Identified

The dataset comprising 583 patient records was sourced from Kaggle, including attributes such as age, gender, and various biochemical parameters relevant to liver disease

diagnosis. Data quality assurance included verifying data integrity and addressing missing values.

3.2. Data Quality Report

Initial data preprocessing involved cleaning the dataset and handling missing values. Specifically, the Albumin_and_Globulin_Ratio column had 4 missing values which were addressed using appropriate imputation techniques.

3.3. Data Preprocessing

Preprocessing steps included:

Encoding categorical variable Gender to numerical values.

- Scaling numerical features to ensure uniformity and compatibility with machine learning algorithms.

4.Model Development Phase

The Model Development Phase entails crafting a predictive model for loan approval. It encompasses strategic feature selection, evaluating and selecting models (Logistic Regression, Random Forest Classifier, KNN, SVC), initiating training with code, and rigorously validating and assessing model performance for informed decision-making in the lending process.

4.1. Model Selection Report

The models selected—Support Vector Classifier (SVC), K Neighbors Classifier (KNN), Logistic Regression, and Random Forest Classifier—were chosen for their effectiveness in binary classification tasks and potential to achieve high predictive accuracy for detecting liver disease.

4.2. Initial Model Training Code, Model Validation and Evaluation Report

Initial model training involved:

Splitting the dataset into training (70%) and testing (30%) sets.
Training each classifier with the training set.
Evaluating model performance using metrics such as accuracy, precision, recall, F1-score, and confusion matrices.

5.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.

5.1. Tuning Documentation

Hyperparameter tuning was performed to optimize each model's performance:

- For Logistic Regression, the best parameters were {'C': 1.0, 'penalty': 'l1', 'solver': 'liblinear'}.
- SVC was tuned with {'C': 1.6601864044243653, 'gamma': 'scale', 'kernel': 'linear'}.
- Random Forest Classifier's optimal parameters were {'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 10, 'n_estimators': 100}.
- K Neighbors Classifier used {'metric': 'manhattan', 'n_neighbors': 3, 'weights': 'distance'}.

5.2. Final Model Selection Justification

The final models were selected based on their tuned performance metrics:

- Logistic Regression achieved an accuracy of 0.697, with a focus on high precision and recall for disease detection.
- SVC and Random Forest Classifier also showed competitive performance after tuning, supporting their suitability for the task.
- K Neighbors Classifier, while slightly less accurate, contributed valuable insights into the dataset's structure and potential feature interactions.

6.Results

6.1. Output Screenshots

```
Best parameters for Logistic Regression: {'C': 1.0, 'penalty': 'l1', 'solver': 'liblinear'}
Accuracy of Logistic Regression:  0.6971428571428572
Classification Report of Logistic Regression:               precision    recall  f1-score   support

           1       0.72      0.93      0.81       122
           2       0.50      0.15      0.23        53

    accuracy                           0.70       175
   macro avg       0.61      0.54      0.52       175
weighted avg       0.65      0.70      0.64       175

Cross Validation Score of Logistic Regression:  0.7352941176470589
Confusion Matrix of Logistic Regression:  [[114  45]
 [  8   8]]
Support Vector Classifier (SVC)...
Fitting 3 folds for each of 10 candidates, totalling 30 fits
Best parameters for SVC: {'C': np.float64(1.6601864044243653), 'gamma': 'scale', 'kernel': 'linear'}
Accuracy of SVC:  0.6971428571428572
Classification Report of SVC               precision    recall  f1-score   support

           1       0.70      1.00      0.82       122
           2       0.00      0.00      0.00        53

    accuracy                           0.70       175
   macro avg       0.35      0.50      0.41       175
weighted avg       0.49      0.70      0.57       175
```

```
Cross Validation Score of SVC:  0.7181372549019608
Confusion Matrix of SVC:  [[122  53]
 [  0   0]]
Random Forest Classifier...
Fitting 5 folds for each of 108 candidates, totalling 540 fits
Best parameters for Random Forest Classifier: {'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 10, 'n_estimators': 100}
Accuracy of Random Forest Classifier:  0.6742857142857143
Classification Report of Random Forest Classifier:               precision    recall  f1-score   support

         1       0.71      0.89      0.79       122
         2       0.41      0.17      0.24        53

    accuracy                           0.67       175
   macro avg       0.56      0.53      0.52       175
weighted avg       0.62      0.67      0.63       175

Cross Validation Score of Random Forest Classifier:  0.7181372549019608
Confusion Matrix of Random Forest Classifier:  [[109  44]
 [ 13   9]]
K Neighbors Classifier...
Fitting 5 folds for each of 16 candidates, totalling 80 fits
Best parameters for K Neighbors Classifier: {'metric': 'manhattan', 'n_neighbors': 3, 'weights': 'distance'}
Accuracy of KNN:  0.6628571428571428
```

```
Classification Report of KNN:               precision    recall  f1-score   support

           1       0.74      0.80      0.77       122
           2       0.43      0.36      0.39        53

    accuracy                           0.66       175
   macro avg       0.59      0.58      0.58       175
weighted avg       0.65      0.66      0.65       175

Cross Validation Score of KNN:  0.7181372549019609
Confusion Matrix of KNN:  [[97 34]
 [25 19]]

Process finished with exit code 0
```

7.Advantages & Disadvantages

Advantages of Using Machine Learning for Liver Patient Analysis

1. **Early Detection**: Machine learning (ML) algorithms can identify patterns and anomalies in medical data that may indicate liver disease at an early stage, leading to earlier diagnosis and treatment.
2. **Improved Accuracy**: ML models can analyze vast amounts of data with high accuracy, reducing the chances of human error in diagnosing liver conditions.
3. **Personalized Treatment**: ML can tailor treatment plans based on individual patient data, optimizing therapy and improving outcomes.
4. **Predictive Analytics**: ML can predict disease progression and patient outcomes, helping in planning long-term care and management.
5. **Resource Efficiency**: Automated analysis using ML can save time and resources for healthcare providers, allowing them to focus on patient care.
6. **Data Integration**: ML can integrate and analyze data from multiple sources (e.g., imaging, lab results, patient history), providing a comprehensive view of the patient's health.

Disadvantages of Using Machine Learning for Liver Patient Analysis

1. **Data Quality**: ML models rely on the quality and quantity of data; poor or biased data can lead to inaccurate results.
2. **Interpretability**: Some ML models, especially deep learning, can be complex and difficult to interpret, making it hard for clinicians to understand the reasoning behind certain predictions.
3. **Cost**: Developing and maintaining ML systems can be expensive, requiring significant investment in technology and expertise.

4. **Privacy Concerns**: Handling sensitive medical data with ML raises concerns about patient privacy and data security.
5. **Regulatory Challenges**: Ensuring compliance with healthcare regulations and obtaining approvals for ML applications can be challenging and time-consuming.
6. **Dependency on Technology**: Over-reliance on ML systems may reduce the emphasis on clinical expertise and human judgment, potentially impacting patient care quality.

By balancing these advantages and disadvantages, healthcare providers can better utilize ML to improve liver patient analysis while addressing potential challenges

8.Conclusion

Initially, the dataset was explored and made ready to be fed into the classifiers. This was achieved by removing some rows containing null values, transforming some columns which were showing skewness and using appropriate methods (one-hot encoding or label encoder) to convert the labels so that they can be useful for classification purposes. Performance metrics on which the models would be evaluated were decided. The dataset was then split into a training and testing set.

Firstly, a naive predictor and a benchmark model ('Logistic Regression') were run on the dataset to determine the benchmark value of accuracy. The greatest difficulty in the execution of this project was faced in two areas- determining the algorithms for training and choosing proper parameters for fine-tuning. Initially, I found it very vexing to decide upon 3 or 4 techniques out of the numerous options available in sklearn.

This exercise made me realize that parameter tuning is not only a very interesting but also a very important part of machine learning. I think this area can warrant further improvement, if we are willing to invest a greater amount of time as well as computing power.

9.Future Scope

Future Scope of Liver Patient Analysis Using Machine Learning

1. **Enhanced Diagnostic Accuracy**: Future advancements in ML algorithms could significantly improve the precision of liver disease diagnosis, identifying subtle patterns and anomalies that are currently undetectable.
2. **Real-Time Monitoring**: Integration of ML with wearable devices and IoT technology could enable continuous real-time monitoring of liver health, providing immediate alerts for any concerning changes.

3. **Predictive Maintenance**: ML could predict the likelihood of disease progression or relapse, allowing for proactive and preventive measures to be taken, ultimately improving patient outcomes.
4. **Personalized Medicine**: Advances in ML could lead to highly personalized treatment plans based on a patient's genetic makeup, lifestyle, and medical history, optimizing therapeutic strategies for each individual.
5. **Drug Development**: ML could accelerate drug discovery and development for liver diseases by identifying potential drug candidates and predicting their effectiveness and side effects through advanced modeling techniques.
6. **Integration with Genomics**: Combining ML with genomic data could uncover genetic markers associated with liver diseases, leading to a deeper understanding of disease mechanisms and the development of targeted therapies.
7. **Healthcare Accessibility**: ML applications could make advanced liver disease analysis accessible to remote and underserved areas, improving global health equity by providing high-quality diagnostic tools regardless of location.
8. **Automated Reporting**: Future ML systems could automatically generate detailed and accurate reports from complex medical data, reducing the burden on healthcare professionals and minimizing human error.
9. **Enhanced Imaging Analysis**: ML could further enhance the analysis of medical imaging, such as CT and MRI scans, providing more detailed and accurate assessments of liver condition and aiding in early detection of liver abnormalities.

The future of liver patient analysis using ML is promising, with potential to revolutionize how liver diseases are diagnosed, monitored, and treated, leading to improved patient care and outcomes.

10.Appendix

10.1. Source Code :

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import pickle
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV, RandomizedSearchCV
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.exceptions import UndefinedMetricWarning
import warnings
from scipy.stats import uniform
```

```
warnings.filterwarnings("ignore", category=FutureWarning)
warnings.filterwarnings('ignore', category=UndefinedMetricWarning)
#data = pd.read_csv(r"https://www.kaggle.com/datasets/uciml/indian-liver-patient-records")
# download the dataset using the above link and copy paste the link here
data = pd.read_csv(r"C:\Users\VAISHNAVI\OneDrive\Desktop\ML datasets\indian_liver_patient.csv")
# download the dataset using the above link and copy paste the link here
print(data)
print(data.info())
print(data.describe())
print(data.columns)
print(data.isnull().sum())

print(data.shape)
data['Albumin_and_Globulin_Ratio'] =
data['Albumin_and_Globulin_Ratio'].fillna(data['Albumin_and_Globulin_Ratio'].mode()[0])
print(data.isnull().sum())
sns.countplot(x=data['Gender'], data=data)
m, f=data['Gender'].value_counts()
print("No. of Males: ", m)
print("No. of Females: ", f)
plt.show()
sns.countplot(x=data['Dataset'], data=data)
LD, NLD=data['Dataset'].value_counts()
print("Liver disease patients: ", LD)
print("Non-Liver disease patients: ", NLD)
plt.show()
sns.set_style('darkgrid')
plt.figure(figsize=(25,10))
data['Age'].value_counts().plot.bar(color='darkviolet')
plt.show()
f, ax = plt.subplots(figsize=(8, 6))
sns.scatterplot(x="Albumin", y="Albumin_and_Globulin_Ratio",color='mediumspringgreen',data=data);
plt.show()
sns.pairplot(data)
plt.show()
sns.boxplot(data['Albumin_and_Globulin_Ratio'])
plt.show()
x = data.iloc[:, 0:-1]
y = data.iloc[:, -1]
print(x)
print(y)
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=0, test_size=0.3)
le = LabelEncoder()
x_train_gender = le.fit_transform(x_train['Gender'])
```

```
x_test_gender = le.transform(x_test['Gender'])
print(le.classes_)
x_train_gender =x_train_gender.reshape(-1, 1)
x_test_gender = x_test_gender.reshape(-1, 1)
x_train = x_train.drop('Gender', axis=1)
x_test = x_test.drop('Gender', axis=1)
x_train_combined = np.concatenate((x_train.values, x_train_gender), axis=1)
x_test_combined = np.concatenate((x_test.values, x_test_gender), axis=1)
column_names = list(x_train.columns) + ['Gender']
x_train_final = pd.DataFrame(x_train_combined, columns=column_names)
x_test_final = pd.DataFrame(x_test_combined, columns=column_names)

print("Shape of x_train_combined:", x_train_final.shape)
print("Shape of x_test_combined:", x_test_final.shape)

print(x_train_final.sample(5))
x_train_final.to_csv('x_train_final.csv', index=False)
x_test_final.to_csv('x_test_final.csv', index=False)
y_test.to_csv('y_test.csv', index=False)
print("Logistic Regression...")
lr_param_grid = {
    'penalty': ['l1', 'l2'],
    'C': [0.001, 0.01, 0.1, 1.0, 10.0],
    'solver': ['liblinear', 'saga']
}
lr_s = LogisticRegression(max_iter=1000)
lr_grid_search = GridSearchCV(estimator=lr_s, param_grid=lr_param_grid, cv=5, scoring='accuracy', verbose=1,
n_jobs=-1)
lr_grid_search.fit(x_train_final, y_train)
# Get best parameters
lr_best_params = lr_grid_search.best_params_
print("Best parameters for Logistic Regression:", lr_best_params)
lr = LogisticRegression(**lr_best_params, max_iter=1000)
lr.fit(x_train_final, y_train)
y_pred_lr = lr.predict(x_test_final)
lr_acc = accuracy_score(y_pred_lr, y_test)
print("Accuracy of Logistic Regression: ", lr_acc)
print("Classification Report of Logistic Regression: ", classification_report(y_test, y_pred_lr))
lr_cross = cross_val_score(lr, x_train_final, y_train, scoring='accuracy', cv = 6)
print("Cross Validation Score of Logistic Regression: ", lr_cross.mean())
lr_cm = confusion_matrix(y_pred_lr,y_test)
print("Confusion Matrix of Logistic Regression: ", lr_cm)
print("Support Vector Classifier (SVC)...")
svm_param_dist = {
```

```
    'C': uniform(0.1, 10),
    'kernel': ['linear', 'rbf'],
    'gamma': ['scale', 'auto']
}
svm_s = SVC()
# Initialize GridSearchCV
svm_random_search = RandomizedSearchCV(estimator=svm_s, param_distributions=svm_param_dist, n_iter=10,
cv=3, scoring='accuracy', verbose=1, n_jobs=2, random_state=42)
# Fit GridSearchCV
svm_random_search.fit(x_train_final, y_train)

svm_best_params = svm_random_search.best_params_
print("Best parameters for SVC:", svm_best_params)
svm = SVC(**svm_best_params)
svm.fit(x_train_final, y_train)
y_pred_svm = svm.predict(x_test_final)
svm_acc = accuracy_score(y_pred_svm, y_test)
print("Accuracy of SVC: ", svm_acc)
print("Classification Report of SVC", classification_report(y_test, y_pred_svm))
svm_cross = cross_val_score(svm, x_train_final, y_train, scoring='accuracy', cv = 6)
print("Cross Validation Score of SVC: ", svm_cross.mean())
svm_cm = confusion_matrix(y_pred_svm,y_test)
print("Confusion Matrix of SVC: ", svm_cm)


print("Random Forest Classifier...")
rfc_param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
rfc_s = RandomForestClassifier()
rfc_grid_search = GridSearchCV(estimator=rfc_s, param_grid=rfc_param_grid, cv=5, scoring='accuracy',
verbose=1, n_jobs=-1)
rfc_grid_search.fit(x_train_final, y_train)
rfc_best_params = rfc_grid_search.best_params_
print("Best parameters for Random Forest Classifier:", rfc_best_params)
rfc = RandomForestClassifier(**rfc_best_params)
rfc.fit(x_train_final, y_train)
ypred_rfc = rfc.predict(x_test_final)
rfc_acc = accuracy_score(ypred_rfc, y_test)
print("Accuracy of Random Forest Classifier: ", rfc_acc)
print("Classification Report of Random Forest Classifier: ", classification_report(y_test, ypred_rfc))
```

```
rfc_cross = cross_val_score(rfc, x_train_final, y_train, scoring='accuracy', cv = 6)
print("Cross Validation Score of Random Forest Classifier: ", rfc_cross.mean())
rfc_cm = confusion_matrix(ypred_rfc,y_test)
print("Confusion Matrix of Random Forest Classifier: ", rfc_cm)

print("K Neighbors Classifier...")
knn_param_grid = {
    'n_neighbors': [3, 5, 7, 9],
    'weights': ['uniform', 'distance'],
    'metric': ['euclidean', 'manhattan']
}

knn_s = KNeighborsClassifier()
knn_grid_search = GridSearchCV(estimator=knn_s, param_grid=knn_param_grid, cv=5, scoring='accuracy',
verbose=1, n_jobs=-1)
# Fit GridSearchCV
knn_grid_search.fit(x_train_final, y_train)

knn_best_params = knn_grid_search.best_params_
print("Best parameters for K Neighbors Classifier:", knn_best_params)
knn = KNeighborsClassifier(**knn_best_params)
knn.fit(x_train_final, y_train)
ypred_knn = knn.predict(x_test_final)
knn_acc = accuracy_score(ypred_knn, y_test)
print("Accuracy of KNN: ", knn_acc)
print("Classification Report of KNN: ", classification_report(y_test, ypred_knn))
knn_cross = cross_val_score(knn, x_train_final, y_train, scoring='accuracy', cv = 6)
print("Cross Validation Score of KNN: ", knn_cross.mean())
knn_cm = confusion_matrix(ypred_knn,y_test)
print("Confusion Matrix of KNN: ", knn_cm)
pickle.dump(svm, open('svm_liver_analysis.pkl', 'wb'))
pickle.dump(rfc, open('rfc_liver_analysis.pkl', 'wb'))
pickle.dump(knn, open('knn_liver_analysis.pkl', 'wb'))
pickle.dump(lr, open('lr_liver_analysis.pkl', 'wb'))
```

Model Deployment code

```
from flask import Flask, render_template, request
import pickle
import numpy as np

app = Flask(__name__, template_folder='templates')

# Load models
```

```
svm_model = pickle.load(open('svm_liver_analysis.pkl', 'rb'))
rfc_model = pickle.load(open('rfc_liver_analysis.pkl', 'rb'))
knn_model = pickle.load(open('knn_liver_analysis.pkl', 'rb'))
lr_model = pickle.load(open('lr_liver_analysis.pkl', 'rb'))

@app.route('/')
def loadpage():
    return render_template("LiverPredict.html")

@app.route('/y_predict', methods=["POST"])
def prediction():
    try:
        # Extracting input values from the form

        Gender = request.form.get("Gender")
        Age = request.form.get("Age")
        Total_Bilirubin = request.form.get("Total_Bilirubin")
        Direct_Bilirubin = request.form.get("Direct_Bilirubin")
        Alkaline_Phosphotase = request.form.get("Alkaline_Phosphotase")
        Alamine_Aminotransferase = request.form.get("Alamine_Aminotransferase")
        Aspartate_Aminotransferase = request.form.get("Aspartate_Aminotransferase")
        Total_Proteins = request.form.get("Total_Proteins")
        Albumin = request.form.get("Albumin")
        Albumin_and_Globulin_Ratio = request.form.get("Albumin_and_Globulin_Ratio")

        # Debugging: print received form data
        print(f"Received data: Gender={Gender}, Age={Age}, Total_Bilirubin={Total_Bilirubin},
Direct_Bilirubin={Direct_Bilirubin}, Alkaline_Phosphotase={Alkaline_Phosphotase}, "
              f"Alamine_Aminotransferase={Alamine_Aminotransferase},
Aspartate_Aminotransferase={Aspartate_Aminotransferase}, Total_Proteins={Total_Proteins},
Albumin={Albumin},"
              f" Albumin_and_Globulin_Ratio={Albumin_and_Globulin_Ratio}")

        # Encoding gender
        gender_dict = {'Male': 1, 'Female': 0}
        Gender_encoded = gender_dict.get(Gender)

        # Check if Gender is valid
        if Gender_encoded is None:
            return render_template("LiverPredict.html", prediction_text="Invalid Gender input")

        # Creating a data array for prediction
        data = [[float(Gender_encoded), float(Age), float(Total_Bilirubin), float(Direct_Bilirubin),
                 float(Alkaline_Phosphotase), float(Alamine_Aminotransferase),
```

```
            float(Aspartate_Aminotransferase), float(Total_Proteins), float(Albumin),
            float(Albumin_and_Globulin_Ratio)]]
        data = np.array(data).reshape(1, -1)

        # Choose model to predict (example: SVM)
        model = svm_model
        prediction = model.predict(data)[0]

        # Generating prediction result text
        if prediction == 1:
            result_text = "You have a liver disease problem, you should see a doctor."
        else:
            result_text = "You do not have a liver disease problem."

        return render_template("LiverPredict.html", prediction_text=result_text)

    except Exception as e:
        return render_template("LiverPredict.html", prediction_text=f"Error: {str(e)}")

if __name__ == "__main__":
    app.run(debug=True)
```

10.2. GitHub & Project Demo Link

GitHub : https://github.com/VaishnaviMoharana/Prediction-and-Analysis-of-Liver-Patient-Data-Using-Machine-Learning.git

Project Demo Link : https://drive.google.com/drive/folders/1vAKsxSQTvJO-7xbJyIws7KDNVE9N29CL?usp=sharing