ADVANCEMACHINGLEARNINGPROJECT

1. Introduction

- **ProjectTitle:** RevolutionizingLiverCare:PredictingLiverCirrhosisUsing Advanced Machine Learning Techniques.

- **TeamMembers:** 1) Chennamsetty Guruteja

  2) Cherivi Suryavardhan

  3) Doddipalli Venkata Siva Reddy

  4) G R Tejaswini

2. ProjectOverview

- **Purpose:** This project aims to develop a machine learning-based predictive model forearlydetectionoflivercirrhosisusingclinicalandbiochemicaldata.Byapplying advanced algorithms and performance tuning, the model identifies patients at risk with high accuracy. The best- performing model is deployed using Flask, enabling integrationintoreal-timediagnosticsystems.Thisapproachsupportsfaster,data-driven decisions in liver healthcare and improves patient outcomes through early intervention.

- **Features**:TheLiverCirrhosisPredictionprojectisacomprehensivemachine learningsolutiondesignedtopredictthelikelihoodoflivercirrhosisinpatientsbased on their medical and lifestyle data. The project involves key steps such as data preprocessing,handlingmissingvalues,encodingcategoricalvariables,and normalizingnumericalfeatures.MultipleclassificationalgorithmsincludingLogistic Regression, Random Forest, K-Nearest Neighbors (KNN), and XGBoost are trained and evaluated using metrics like accuracy, precision, recall, and F1 score. The best-performing model is selected, saved, and integrated into a Flask-based web application,enablinguserstoinputpatientdataandreceiveinstantpredictions.This projectaimstosupportearlydiagnosisandassisthealthcareprofessionalsinclinical decision-making.

3. Architecture

- **Frontend:**ThefrontendoftheLiverCirrhosisPredictionprojectisasimple,user-friendly interface built with HTMLand CSS. It includes a form where users input patientdetailslikeage,gender,medicalhistory,andtestresults.Uponsubmission,the data is sent to

the Flask backend, which returns a prediction. The result is then displayedclearlyonthesamepage,makingiteasyforuserstounderstandwhether the patient is at risk of liver cirrhosis.

- **Backend:**ThebackendoftheLiverCirrhosisPredictionprojectisbuiltusingFlask, a lightweight Python web framework. It handles user input from the frontend, processesthedata, andloadsatrainedmachinelearning modeltomakepredictions. Thebackend alsoincludesdatapreprocessing stepslikeencodingand scaling to ensuretheinputmatchesthemodel'srequirements.Oncethepredictionisgenerated, the result is sent back to the frontend for display. The backend ensures smooth integrationbetweentheuserinterfaceandthepredictivemodel,maintainingaccuracy and performance.

- **Database:**TheprojectusesMongoDBtostorepatientinputdataandcorresponding liver cirrhosis prediction results. Each prediction is saved as a document containing keymedicalfeatures,the model'soutcomeandatimestamp.Whenausersubmits datathroughthefrontend,thebackendprocessestheinput,predictstheresultusinga trained model, and stores the full record in the predictions collection using insert_one(). MongoDB provides flexible schema design, seamless integration with Flask using PyMongo, and efficient storage of medical records for further analysis.

4. SetupInstructions

- **Prerequisites:**

1. Python3.x–Corelanguagefordevelopment
2. Pandas– Fordata manipulationand preprocessing
3. NumPy–Fornumericaloperations
4. Scikit-learn–Machinelearningmodelsandevaluationmetrics
5. XGBoost– Gradientboosting classifier
6. Flask–Backendwebframework
7. Flask-PyMongo–ToconnectFlask withMongoDB
8. PickleorJoblib–For saving andloading ML models
9. MongoDB–NoSQLdatabasetostoreuserinputsand predictions

- **Installation:**

1. Python3installed
2. Flaskinstalled:pipinstallflask
3. Clone the repository: https://github.com/CHENNAMSETTYGURUTEJA/Liver_Cirrhosis_Prediction
4. Navigatetoprojectfolder:
   cd liver_cirrhosis
   cdproject_executable_files
5. RuntheFlaskapp:pythonapp.py

6. Visit: http://127.0.0.1:5000/

5. Folder Structure

- **Client**: In this project, the client refers to the frontend interface through which users interact with the liver cirrhosis prediction system. The client is built using HTML, CSS, and optionally JavaScript to ensure a responsive and user-friendly experience. Users can input patient data via a structured form, and upon submission, the data is sent to the Flask backend for processing. The client then displays the prediction result (whether the patient is likely suffering from liver cirrhosis or not) clearly on the same or a redirected results page. This interface aims to be intuitive, clean, and accessible for medical professionals or researchers using the system.

- **Server**: The **server** in this project is developed using the Flask web framework, serving as the backend that handles the core logic and communication between the client and the machine learning model. When a user submits input data through the frontend, the Flask server receives the data via HTTP requests, preprocesses it as needed, loads the trained machine learning model, and performs the prediction. It then sends the prediction result back to the client for display. Additionally, the server ensures proper routing, error handling, and integrates seamlessly with the MongoDB database for storing input data and prediction results if required.

6. API Documentation

1. POST `/predict`

- **Description**: Accepts patient data and returns a prediction indicating whether the patient is likely to suffer from liver cirrhosis.
- **Request Type**: `application/json`
- **Request Body**:

```
{
 "Age":45,
 "Gender": "male",
 "Duration_of_alcohol_consumption":10,
 "Hepatitis_B_infection": "positive",
 ...
}
```

- **Response**:

```
{
 "prediction":1,
 "result":"Patient is likely suffering from liver cirrhosis"
}
```

2. GET `/`

- **Description**: Returns a welcome message or a homepage route.

- **Response**:

```
{
  "message":"WelcometotheLiverCirrhosisPredictionAPI"
}
```

2. POST`/store-data`*(OptionalifusingMongoDB)*

- **Description**:Savespatientinputandpredictionresulttothedatabase.
- **RequestBody**:Sameas`/predict`
- **Response**:

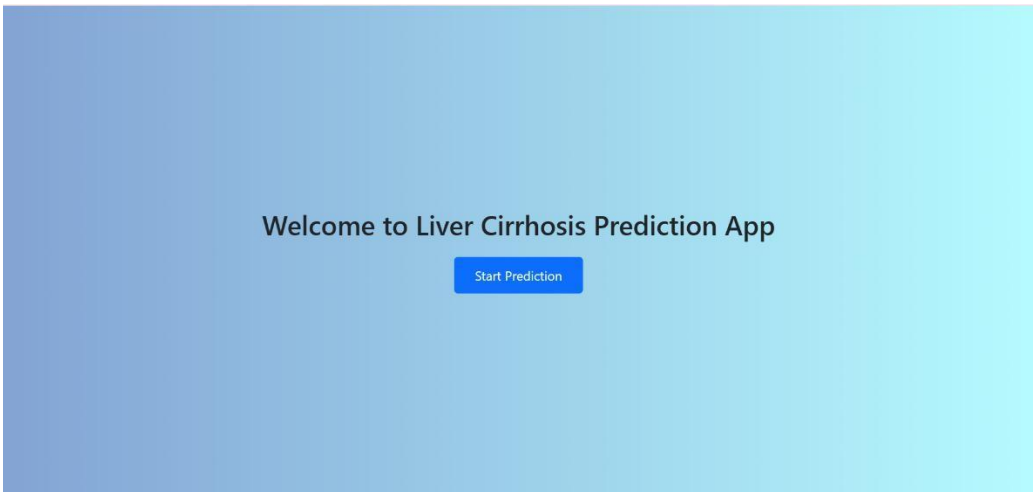
```
{
  "status":"success",
  "message":"Datastoredsuccessfully"
}
```

8. Authentication

- Nospecificloginorauthenticationisprovidedforthisprojectasitcanbeusedby anyone and everyone.

9. UserInterface

- The User Interface (UI) of the Liver Cirrhosis Prediction project is designed to be clean,user-friendly,andaccessibletobothmedicalprofessionalsandnon-technical users. It focuses on simplicity and functionality, enabling users to easily input data and receive predictions.



9. Testing

- To ensure the accuracy, reliability, and robustness of the Liver Cirrhosis Prediction project, a combination of testing strategies and tools were

employedacrossbothmachinelearningandwebapplicationcomponents.
Testing Strategies:

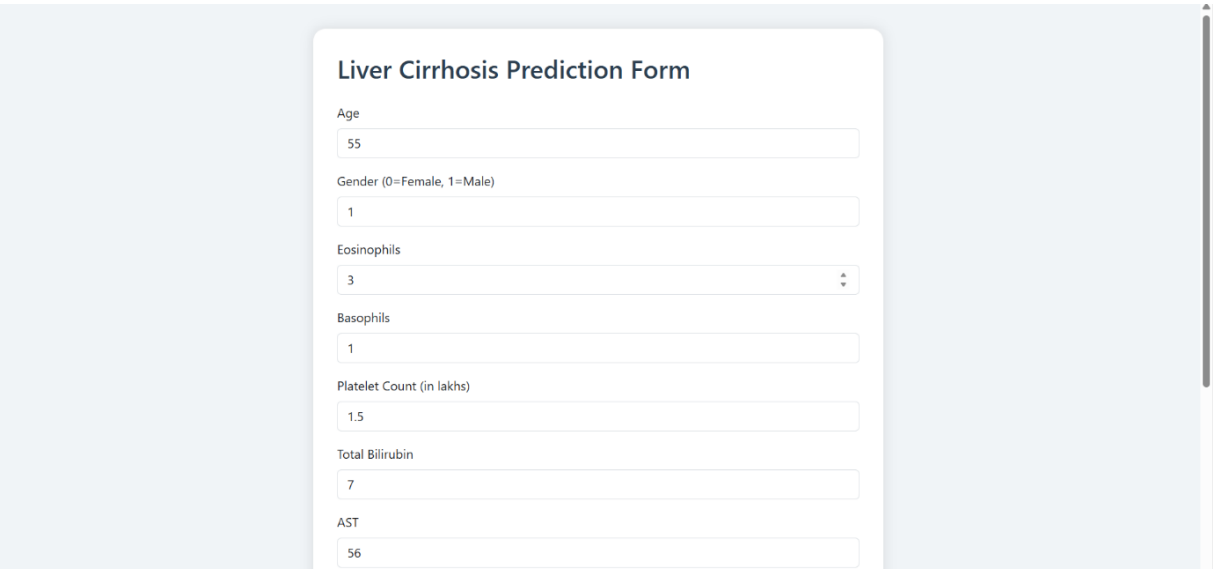
1. **DataValidationTesting:**
    - Checkedformissingvalues,outliers,andincorrectdatatypes.
    - Ensuredbalancedpreprocessing,especiallyforcategorical encoding and numerical normalization.
2. **ModelEvaluation:**
    - Used**stratifiedtrain-testsplit**topreserveclassdistribution.
    - Evaluatedmultiplemodels(LogisticRegression,RandomForest, KNN, XGBoost) using metrics like **accuracy, precision, recall, and F1-score**.
    - Applied**cross-validation**toensuregeneralization.
3. **APITesting:**
    - Ensuredendpointsrespondcorrectlytovalidandinvalidinputs.
    - CheckedforcorrectHTTPstatuscodesandJSONresponses.
4. **IntegrationTesting:**
    - Validatedend-to-endflowfromUItobackendtomodelandback.
    - Confirmedcorrectdatahandlingbetweenfrontendforms,Flask server, and MongoDB database.
5. **EdgeCaseTesting:**
    - Inputedgevalues(e.g.,extremeages,abnormaltestresults)to observe model and system behavior.

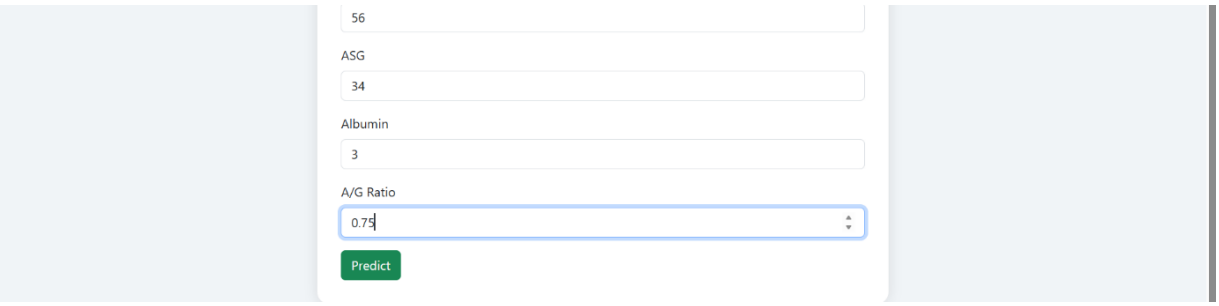
ToolsUsed:

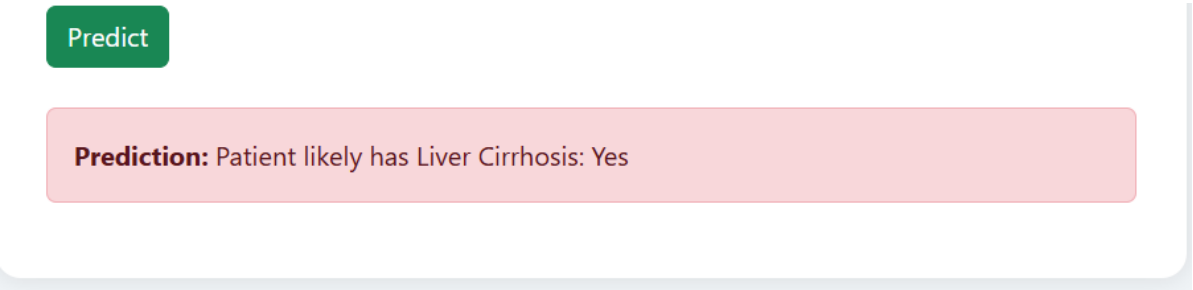
- **Scikit-learn**–formodelevaluationandmetrics.
- **Pytest/Unittest(Python)**–totestbackendfunctionsandlogic.
- **Postman/ cURL** – formanual APItesting.
- **JupyterNotebook** –forexploratorydataanalysis andtestingpreprocessinglogic.

.BrowserDevTools –totestanddebugfrontendinteractions.

10. Screenshots







11 .KnownIssues

WiletheLiverCirrhosisPredictionSystemfunctionseffectivelyinmostcases,thereare several limitations and known issues that users and developers should be aware of:

1. DataImbalance

- Thedatasetisheavilyimbalanced (very fewpositivecirrhosiscases).
- Thismayleadtooverfittingwheremodelspredictthemajorityclassmoreoften, impacting real-world accuracy.
- Mitigation:Stratifiedsplitting,modeltuning,andfutureinclusionofresampling techniques (SMOTE) are recommended.

2. MissingorIncompleteData

- Severalfeaturesintheoriginaldatasetcontainmissingvalues,especiallyinmedical metrics like MCH, RBC, TG, and LDL.
- Mitigation:Imputationisperformedusingmean/mode,butitmayaffectmodel reliability. Future data collection should focus on completeness.

3. PrecisionWarninginClassificationReport

- Incaseswherethemodelpredictsonlyoneclass,scikit-learnraisesawarning: *"Precisionisill-definedandbeingsetto0.0inlabelswithnopredictedsamples..."*
- Mitigation:Ensuringpropermodelgeneralizationandhandlingedgecaseshelps reduce this issue.

4. NoRole-BasedAccessControl

- Thecurrentauthenticationsystemisbasic(login/signup)anddoesnotsupport role- based access (e.g., doctor vs patient).
- Mitigation:Futureversionsshouldimplementuserrolesandpermissionsfor enhanced security.

5. LimitedInputValidationonFrontend

- Thefrontenddoesnotfullyvalidateforminputs(e.g.,negativenumbers,non-numeric entries).
- Mitigation:JavaScript-basedandbackendvalidationshouldbeaddedtoprevent malformed data submissions.

6.BrowserCompatibility

- UImightdisplayinconsistentlyonolderbrowsers ormobileviews.
- Mitigation:Frontendshouldbetestedwithresponsivedesigntoolsandcross-browser testing.

7.ModelDeploymentLimitations

- ModelishostedwithintheFlaskserver andmaynot scalewellforhigh traffic.
- Mitigation:ConsidercontainerizingwithDockeranddeployingviaserviceslike Heroku or AWS for better scalability.

13. FutureEnhancements

- **ModelExplainability**
  IntegrateSHAPorLIME tohelpusersunderstandthereasoningbehindpredictions.
- **Real-TimeAPIIntegration**
  Exposetheprediction modelviaaRESTAPI for real-time usein hospitalsor clinics.
- **Mobile-FriendlyUI**
  Optimizethe frontendfor mobiledevicestoimproveaccessibility.
- **Role-BasedAccessControl**
  Addloginroleslikedoctor,admin,orresearcherforsecure,personalizedaccess.
- **DashboardandVisualization**
  Addavisualdashboardfor trackingpatienttrendsandprediction statistics.