To implement this in a Flask web application, where you upload a CT scan image, and the system predicts the lung disease using your trained model (either Logistic Regression or LightGBM), you will need to:

1. **Set up a Flask app**: This will handle image upload, model inference, and display results.
2. **Load your trained model**: Save your model and load it in the Flask application to make predictions.
3. **Create a webpage with an upload form**: Allow the user to upload an image, send it to the server, process it, and display the predicted disease.

**Step-by-Step Guide:**

**1. Save the Trained Model:**

You need to save the trained models (Logistic Regression and LightGBM) for use in the Flask app.

python

CopyEdit

import joblib

import lightgbm as lgb

# Save Logistic Regression model

joblib.dump(lr\_model, 'lung\_disease\_lr\_model.pkl')

# Save LightGBM model

lgb\_model.save\_model('lung\_disease\_lgb\_model.txt')

**2. Flask App Structure:**

Your project structure should look like this:

graphql

CopyEdit

lung\_disease\_prediction/

│

├── app.py # Main Flask app file

├── lung\_disease\_lr\_model.pkl # Saved Logistic Regression model

├── lung\_disease\_lgb\_model.txt # Saved LightGBM model

└── templates/

└── index.html # HTML page for file upload and result display

└── static/

└── uploads/ # Folder to save uploaded images

**3. Flask App Code (app.py):**

python

CopyEdit

from flask import Flask, request, render\_template

import numpy as np

import joblib

import lightgbm as lgb

from tensorflow.keras.preprocessing.image import load\_img, img\_to\_array

from sklearn.preprocessing import StandardScaler

import os

# Initialize the Flask app

app = Flask(\_\_name\_\_)

# Load the trained models

lr\_model = joblib.load('lung\_disease\_lr\_model.pkl')

lgb\_model = lgb.Booster(model\_file='lung\_disease\_lgb\_model.txt')

# Load LabelEncoder for decoding predictions

import joblib

le = joblib.load('label\_encoder.pkl') # Assuming you saved the encoder

# Image preprocessing parameters

img\_height, img\_width = 128, 128

scaler = StandardScaler()

# Helper function to process images

def process\_image(image\_path):

img = load\_img(image\_path, target\_size=(img\_height, img\_width), color\_mode='grayscale')

img\_array = img\_to\_array(img) / 255.0

img\_array = img\_array.flatten() # Flatten the image for Logistic Regression (if using LR)

img\_array = np.expand\_dims(img\_array, axis=0) # Expand dimensions for prediction

return img\_array

# Route to home page (upload form)

@app.route('/')

def index():

return render\_template('index.html')

# Route to handle image upload and prediction

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

def predict():

if 'file' not in request.files:

return "No file part"

file = request.files['file']

if file.filename == '':

return "No selected file"

if file:

# Save the uploaded file temporarily

file\_path = os.path.join('static/uploads', file.filename)

file.save(file\_path)

# Preprocess the image

img\_array = process\_image(file\_path)

# Normalize the image using StandardScaler (for Logistic Regression)

img\_array\_scaled = scaler.transform(img\_array)

# Make predictions using both models

y\_pred\_lr = lr\_model.predict(img\_array\_scaled)

y\_pred\_lgb = np.argmax(lgb\_model.predict(img\_array), axis=1)

# Decode the labels

predicted\_label\_lr = le.inverse\_transform(y\_pred\_lr)

predicted\_label\_lgb = le.inverse\_transform(y\_pred\_lgb)

# Return the result

return render\_template('index.html', prediction\_lr=predicted\_label\_lr[0], prediction\_lgb=predicted\_label\_lgb[0], img\_path=file\_path)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

**4. HTML Page for Upload Form (templates/index.html):**

Create a simple form to upload an image and display the result.

html

CopyEdit

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Lung Disease Prediction</title>

</head>

<body>

<h1>Lung Disease Prediction</h1>

<form action="/predict" method="POST" enctype="multipart/form-data">

<label for="file">Upload CT Scan Image:</label>

<input type="file" name="file" required><br><br>

<button type="submit">Upload and Predict</button>

</form>

{% if prediction\_lr %}

<h3>Prediction from Logistic Regression: {{ prediction\_lr }}</h3>

{% endif %}

{% if prediction\_lgb %}

<h3>Prediction from LightGBM: {{ prediction\_lgb }}</h3>

{% endif %}

{% if img\_path %}

<h3>Uploaded Image:</h3>

<img src="{{ url\_for('static', filename='uploads/' + file.filename) }}" width="300"><br><br>

{% endif %}

</body>

</html>

**5. Save the Label Encoder:**

To decode the labels (i.e., convert numerical predictions back to their original class names), save the LabelEncoder used during training.

python

CopyEdit

import joblib

# Assuming 'le' is your LabelEncoder

joblib.dump(le, 'label\_encoder.pkl')

**Running the Flask App:**

1. Install the required dependencies:

bash

CopyEdit

pip install flask joblib lightgbm tensorflow scikit-learn

1. Run the Flask app:

bash

CopyEdit

python app.py

1. Visit the page in your browser (usually http://127.0.0.1:5000/), where you can upload a CT scan image and get predictions.