Steps to Create the Library

1. Model Serialization and Loading

Ensure your AI models are saved in a format that can be easily loaded at runtime:

- **TensorFlow/Keras Models:** Save the model as `.h5` or use `SavedModel` format.
- **PyTorch Models:** Save as `.pt` using `torch.save()`.
- **ONNX Models (Optional):** Convert models to ONNX for framework-agnostic usage.

Example for PyTorch:

```
python
Copy code
# Save the model
torch.save(model.state_dict(), "lung_disease_model.pt")

# Load the model
model = LungDiseaseModel()
model.load_state_dict(torch.load("lung_disease_model.pt"))
model.eval()
```

---

2. Preprocessing and Postprocessing Functions

Create standardized preprocessing and postprocessing functions for each type of image (X-ray, MRI, etc.):

```
python
Copy code
import cv2
import numpy as np

def preprocess_image(image_path, target_size=(224, 224)):
    # Load and resize the image
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    image = cv2.resize(image, target_size)
    # Normalize the pixel values
    image = image / 255.0
    image = np.expand_dims(image, axis=(0, -1))  # Add batch and channel
dimensions
    return image

def postprocess_prediction(prediction):
    # Convert raw model output to human-readable format
    if prediction > 0.5:
        return "Disease detected"
    else:
        return "No disease detected"
```

---

3. Wrapper Functions for Models

Create a wrapper function for each model to handle preprocessing, prediction, and postprocessing:

```
python
Copy code
import torch
from models import LungDiseaseModel

def predict_lung_disease(image_path):
    # Load model
    model = LungDiseaseModel()
    model.load_state_dict(torch.load("lung_disease_model.pt"))
    model.eval()

    # Preprocess image
    image = preprocess_image(image_path)
    image_tensor = torch.tensor(image, dtype=torch.float32)

    # Make prediction
    with torch.no_grad():
        output = model(image_tensor)
    prediction = torch.sigmoid(output).item()

    # Postprocess prediction
    result = postprocess_prediction(prediction)
    return result
```

---

4. Create a Python Package

1. **Organize Your Codebase**

   ```
   arduino
   Copy code
   medical_ai/
       models/
           lung_model.py
           blood_cancer_model.py
           ...
       utils/
           preprocess.py
           postprocess.py
       __init__.py
       predict.py
       setup.py
   ```

2. **Write a `setup.py` for Packaging** This allows others to install your library via `pip`.

   ```
   python
   Copy code
   from setuptools import setup, find_packages

   setup(
       name="medical_ai",
       version="1.0.0",
       packages=find_packages(),
       install_requires=[
           "torch",
   ```

```
        "tensorflow",  # Remove if unused
        "opencv-python",
        "numpy"
    ],
    description="AI models for lung issues, blood cancer, eye issues,
and brain tumors",
    author="Your Name",
    author_email="your_email@example.com",
    url="https://github.com/your-repo/medical_ai",
)
```

3. **Build the Package** Run the following commands:

```
bash
Copy code
python setup.py sdist bdist_wheel
```

4. **Distribute the Package**
   - Upload to PyPI using tools like `twine`:

```
bash
Copy code
twine upload dist/*
```

   - Alternatively, host it on a private repository or share via GitHub. **(I prefer lets host it on our git account)**

---

5. Example Usage

Once the library is installed, users can use it like this:

```
python
Copy code
from medical_ai import predict

result = predict.predict_lung_disease("path/to/xray/image.png")
print(result)  # Output: "Disease detected"
```

---

6. Documentation

Provide clear documentation for your library:

- **README.md:** Include usage examples and installation instructions.
- Use tools like **Sphinx** to generate HTML documentation.