Capstone Project: Deployment

Project Title: Medical Diagnostic System for Breast Cancer Detection

Team Members:

1. **Mohammed Adil**
2. **Ali Soltan Dileyta**
3. **Kena Teshome**
4. **Dagim Faji**
5. **Mahdi Abdi Rayaleh**
6. **Impundu Joyeux Thevenin**

1. Overview

In this phase, the breast cancer prediction model is deployed for real-world usage. The deployment process includes making the trained machine learning model accessible to end-users through a web application, allowing them to upload mammography images for classification or use an interactive chatbot to provide symptom details for analysis. The model was integrated into a Django-based backend, with a React frontend for user interaction. This deployment makes it possible for healthcare professionals or patients to use the model for early breast cancer detection.

2. Model Serialization

The trained convolutional neural network (CNN) model was serialized using the **HDF5 format** (`model.h5`). This format is widely used for saving Keras models as it supports efficient storage of large model weights and structures. Serialization ensures that the model's architecture, weights, and configuration can be restored exactly in the production environment. The model is stored as a `.h5` file, which allows for seamless loading and reusability across various platforms without re-training.

3. Model Serving

The serialized model is served using Django, a Python-based web framework. The **`model.h5`** file is loaded in the backend when the web server is initialized. For making predictions, the model is exposed via an API that allows users to send mammography images as input, which are then processed by the model to generate predictions. The Django application serves as the backend for both the image predictor and the chatbot functionalities.

Additionally, **TensorFlow** was used in the backend to load the pre-trained model for real-time predictions, while **React** powers the frontend, ensuring a smooth user experience.

4. API Integration

The deployed model is integrated into a RESTful API for prediction services:

- **Endpoint**: `/api/diagnosis/` (POST request)
- **Input Format**: The API accepts `multipart/form-data` containing the mammography image file (`image` field).
- **Response Format**: The API responds with a JSON object containing the predicted class (`Benign` or `Malignant`) and the confidence score of the prediction.

Example response:

```
json
Copy code
{
  "prediction": "Malignant",
  "confidence": 0.87
}
```

This API is integrated into the React frontend for user interaction, allowing users to submit images and receive predictions.

5. Security Considerations

To secure the deployed model and web application:

- **Authentication**: Access to the admin dashboard is protected by user authentication.
- **CORS**: Cross-Origin Resource Sharing (CORS) is implemented to allow specific frontend URLs to access the API, preventing unauthorized access.
- **HTTPS**: For production, HTTPS is recommended to encrypt data transmission between users and the server.
- **File Validation**: The application checks for valid image files before processing, minimizing the risk of malicious file uploads.

6. Monitoring and Logging

To monitor the performance of the deployed model and the web application:

- **Logging**: Logs are collected for each prediction request, including the image file name, prediction result, confidence score, and timestamp.
- **Performance Metrics**: Key metrics such as accuracy, loss, and model latency are tracked during testing to ensure optimal performance.
- **Error Tracking**: Any prediction or system errors are logged for debugging and improving model performance.
- **Alerts**: If certain error thresholds are met, alerts are sent to the development team for immediate investigation.