Synopsis

INDEX

Title of the Project

Automated Wrist Fracture Detection

by : Akanksha Dhami (00504092024) & Manya Anand (0410409024)

Problem Definition

Wrist fractures are one of the most frequent orthopedic injuries, particularly in elderly and active individuals. Conventional diagnosis via X-ray imaging is subject to human interpretation, which can lead to misdiagnosis and variability in results. This project aims to automate wrist fracture detection using deep learning techniques, leveraging the MURA (Musculoskeletal Radiographs) dataset. By employing convolutional neural networks (CNNs), the system seeks to enhance accuracy, speed, and reliability in detecting fractures, reducing dependency on manual radiological analysis.

Introduction

Medical imaging plays a crucial role in diagnosing fractures, but traditional methods heavily rely on radiologists, whose interpretations can vary due to experience and workload. With the rise of artificial intelligence (AI), deep learning models have demonstrated remarkable success in medical diagnostics by automating image analysis and enhancing detection accuracy.

This project focuses on developing an AI-driven system for wrist fracture detection using the MURA dataset, one of the largest publicly available musculoskeletal radiograph datasets. By training convolutional neural networks (CNNs) on labeled wrist X-ray images, the model will be capable of classifying fractures with high precision. The implementation of this system aims to assist radiologists in making faster, more accurate diagnoses, ultimately improving patient care and treatment outcomes. Additionally, the project seeks to provide an intuitive interface that allows easy integration into clinical workflows, ensuring practical usability in real-world medical settings.

Aim & Objectives

The primary aim is to automate the detection of wrist fractures using deep learning to support radiologists and reduce diagnostic delays.

Objectives:

- Preprocess wrist X-ray images using MURA dataset (`train_labeled_studies.csv` & `valid_labeled_studies.csv`).
- Apply transfer learning with DenseNet121 for feature extraction.
- Train and validate the CNN model.
- Evaluate using accuracy, precision, recall, F1-score.
- Deploy using a **Streamlit-based web interface**.
- Provide a scalable solution adaptable to other fracture types.

Project Category

Machine Learning (Deep Learning - Medical Image Classification):

Machine Learning (ML) is a specialized branch of Artificial Intelligence (AI) that enables systems to automatically learn and improve from data without explicit programming. In this project, we utilize Deep Learning techniques to perform medical image classification, specifically focusing on detecting wrist fractures.

Features of Machine Learning:

Machine Learning introduces advanced concepts over traditional programming techniques, enabling intelligent systems to process large-scale data efficiently. Key characteristics relevant to this project include:

Supervised Learning

Feature Extraction

Model Training and Fine-tuning

Classification and Prediction

Evaluation and Optimization

Automation and Scalability

Dataset:

The project utilizes the MURA v1.1 dataset, specifically the Wrist subset. This subset contains thousands of labeled wrist X-ray images categorized as normal (without fracture) and abnormal (with fracture).

By focusing solely on the wrist data, the model is optimized for wrist fracture detection, ensuring more accurate results compared to training on the entire dataset.

Supervised Learning Approach:

We apply Supervised Learning by training the model using wrist X-ray images labeled as normal or abnormal. The model learns to differentiate between healthy and fractured wrists based on these labeled examples.

Deep Learning Model & Feature Extraction:

The project employs a pre-trained Convolutional Neural Network (CNN) architecture, specifically DenseNet-121 or ResNet-50, which has proven effective in medical image analysis. These models automatically extract crucial features such as bone structure, texture, and fracture patterns from the wrist X-rays.

Transfer Learning:

Using Transfer Learning, the pre-trained model is fine-tuned on the wrist dataset. This significantly improves model accuracy while reducing training time, as the model leverages knowledge gained from large datasets like ImageNet.

Prediction and Classification:

The trained model performs binary classification:

0 → Normal Wrist (No Fracture)

1 → Abnormal Wrist (Fracture Present)

Evaluation Metrics:

The model is evaluated based on:

- Accuracy = TP+TN / TP+TN+FP+FN
- Precision = TP / TP+FP
- Recall = TP / TP+FN
- F1-Score = 2× Precision×Recall / Precision+Recall
- Confusion Matrix

These metrics ensure the reliability and robustness of the wrist fracture detection system.

Automation and Practical Use:

Once deployed, the system provides automated, fast, and accurate diagnosis of wrist fractures, reducing the workload of radiologists and improving early fracture detection.

Reusability and Scalability:

The trained wrist fracture detection model is designed to be:

Reusable across different platforms (web, mobile, hospital systems).

Scalable, allowing future extension to detect fractures in other body parts (e.g., shoulder, elbow) by re-training the same architecture with corresponding data.

Security and Data Protection:

Patient confidentiality is maintained by processing de-identified X-ray images. No personal information is exposed, ensuring compliance with medical data privacy standards.

Dataset Details

The project utilizes the **MURA dataset**, which contains labeled X-ray images of different musculoskeletal structures, including the wrist.

Key Data Insights:

- **Training Data:** Contains labeled wrist X-ray images stored in train_labeled_studies.csv.
- **Validation Data:** Includes separate wrist X-ray images for testing the model's generalization.
- **Labels:** Binary classification (1 = Fracture, 0 = No Fracture).
- **Data Format:** Image file paths linked to corresponding labels.

Tools & Platforms

- **Python**:
  Python was used as the primary programming language for developing the wrist fracture detection system. It is preferred due to its simplicity, readability, and rich ecosystem of machine learning and data science libraries.

- **Google Colab**:
  Google Colab provided a cloud-based environment with free access to GPUs, which significantly accelerated the training process of the deep learning model. It also allowed easy integration with Google Drive for storing datasets and trained models.

- **TensorFlow/Keras**:
  TensorFlow along with its high-level API, Keras, was used to implement and train the DenseNet121 deep learning model. These frameworks offer pre-built modules, layers, and utilities that simplify the construction of complex neural networks.

- **Pandas & NumPy**:
  Pandas and NumPy libraries were used for efficient data manipulation, preprocessing, and handling of CSV files. They provided powerful data structures like DataFrames and arrays, which made it easier to clean and process the MURA dataset.

- **Matplotlib & Seaborn**:
  For visualizing data distributions, training progress, and evaluation metrics, Matplotlib and Seaborn libraries were used. These libraries helped in plotting graphs, heatmaps, and other visualizations to better understand the model’s behavior.

- **Scikit-learn (Sklearn)**:
  Scikit-learn was utilized for model evaluation. Metrics such as accuracy, confusion matrix, precision, recall, and F1-score were calculated using this library to assess the model's performance.

- **DenseNet121**:
  DenseNet121, a pre-trained convolutional neural network model, was used for feature extraction and classification. Its densely connected layers improve information flow and reduce the number of parameters, making it ideal for medical image analysis.

- **Streamlit**:
  Streamlit was employed to develop an interactive web-based interface for the deployment of the wrist fracture detection model. It allowed users to upload X-ray images and view real-time predictions in a user-friendly format.

- **Google Drive**:
  Google Drive was used to store datasets, trained models, and output files securely. Integration with Google Colab made it easy to load data and save results directly to the cloud.

- **MURA v1.1 Dataset (Wrist Subset)**:
  The project utilized the wrist subset of the MURA v1.1 dataset. This dataset contains labeled wrist X-ray images, specifically curated for training and validating the fracture detection model

System Analysis

Existing System:

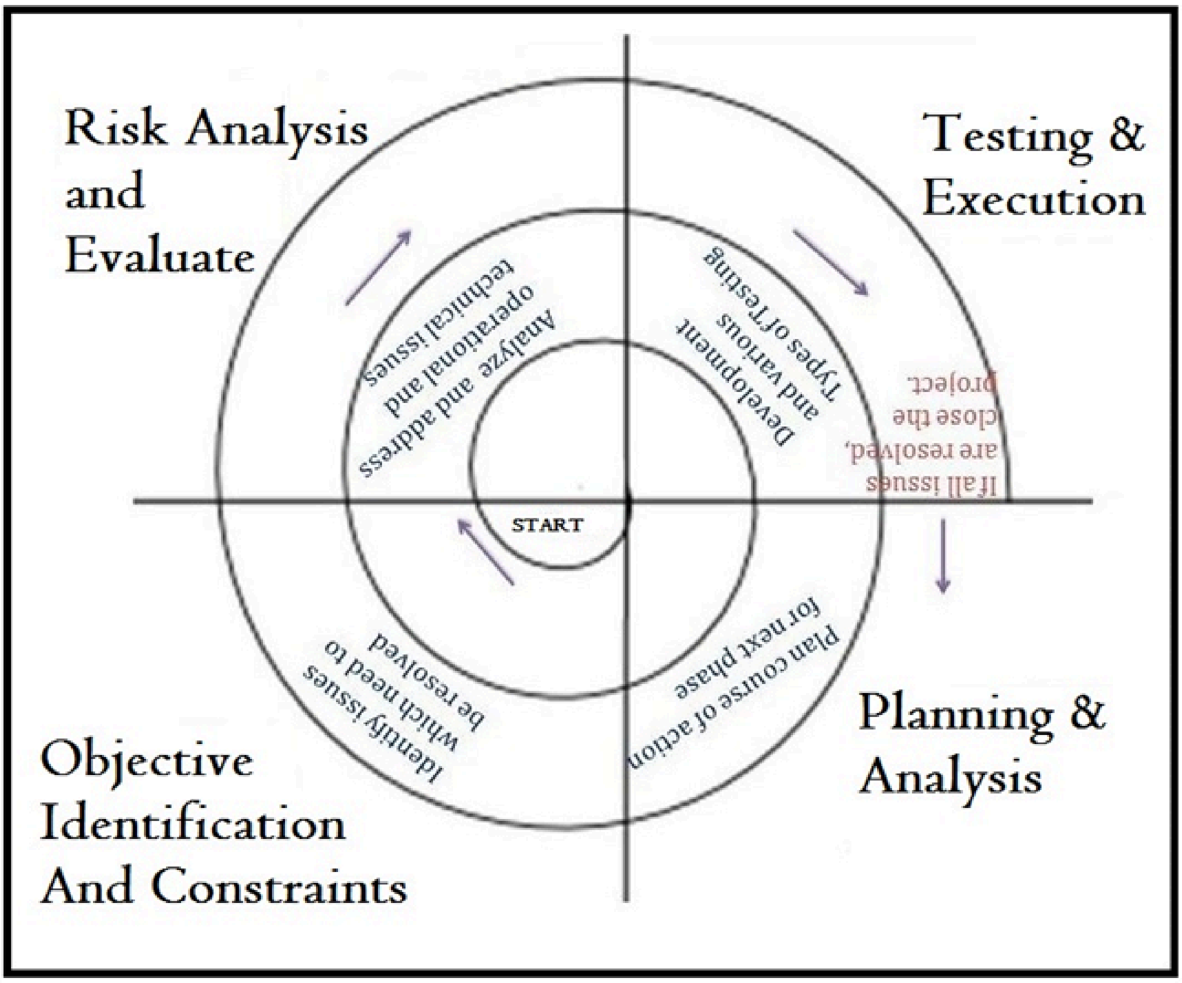
- Manual interpretation of X-rays by radiologists.
- High variability in diagnoses between practitioners.
- Delayed processing in high-patient-load scenarios.

Proposed System:

- A deep learning model trained to classify wrist fractures automatically.
- Reduced diagnostic errors and increased efficiency.
- Deployment in hospitals for real-time fracture detection.

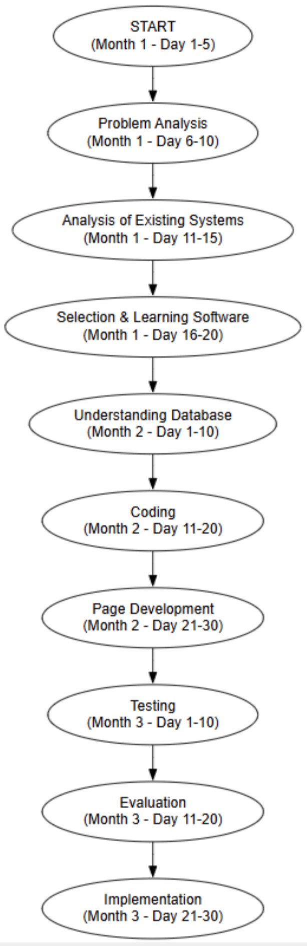
Model Architecture

- A Software requirements specification (SRS), a requirements specification for a software system, is a complete description of the behaviour of a system to be developed and may include a set of use cases that describe interactions the users will have with the software. In addition it also contains non-functional requirements. Non-functional requirements impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints) .
- The software requirements specification document enlists all necessary requirements that are required for the project development.[1] To derive the requirements we need to have clear and thorough understanding of the products to be developed. This is prepared after detailed communications with the project team and customer.
- The SRS may be one of a contract deliverable Descriptions or have other forms of organizationally-mandated content.
- Software Approach
- The spiral model combines the idea of iterative development with the systematic, controlled aspects of the waterfall model.
- Spiral model is a combination of iterative development process model and sequential linear development model i.e. waterfall model with very high emphasis on risk analysis.
- It allows for incremental releases of the product, or incremental refinement through each iteration around the spiral.
- · In each iteration of the spiral approach, software development process follows the phase-wise linear approach. At the end of first iteration, the customer evaluates the software and provides the feedback. Based on the feedback, software development process enters into the next iteration and subsequently follows the linear approach to implement the feedback suggested by the customer. The process of iteration continues throughout the life of the software.


Risk Analysis and Evaluate
Testing & Execution
Analyze and address operational and technical issues
Development and various Types of Testing
If all issues are resolved, close the project.
START
Identify issues which need to be resolved
Plan course of action for next phase
Objective Identification And Constraints
Planning & Analysis

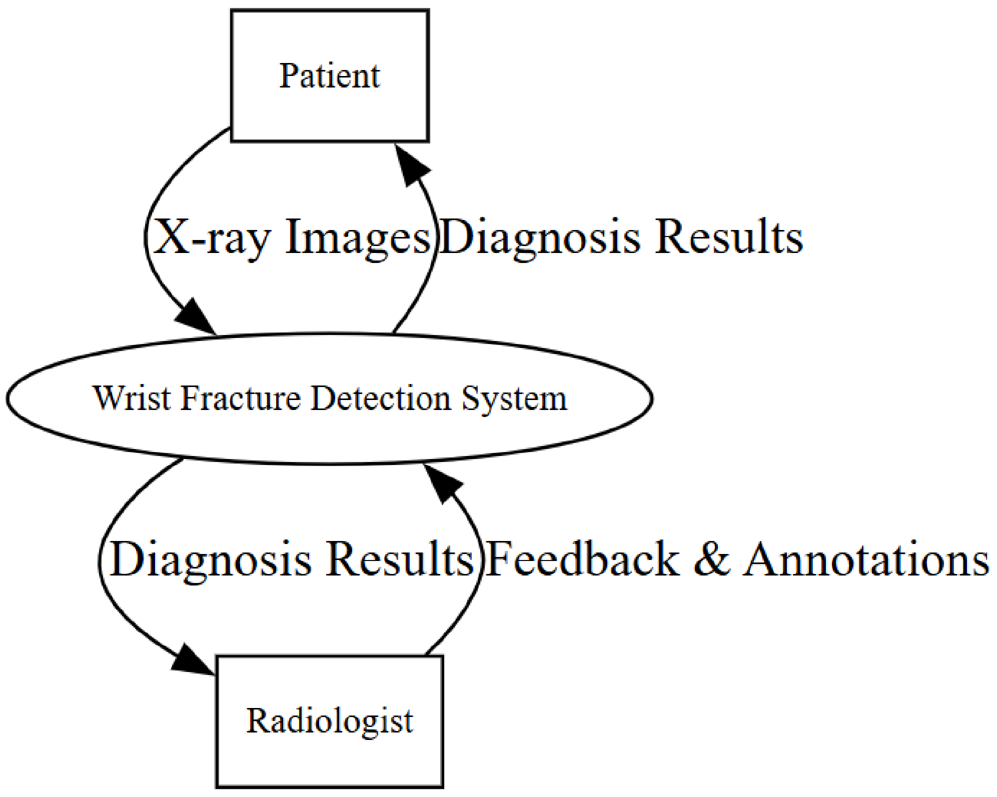
Scheduling

A PERT chart is a type of chart used for project management. It represents each task of a project as a box (circle or rectangle), arrows between tasks are used to show the sequence and task dependencies (i.e. which tasks need to be complete before others can start). Also known as a Network Chart, a precedence diagram and logic diagram

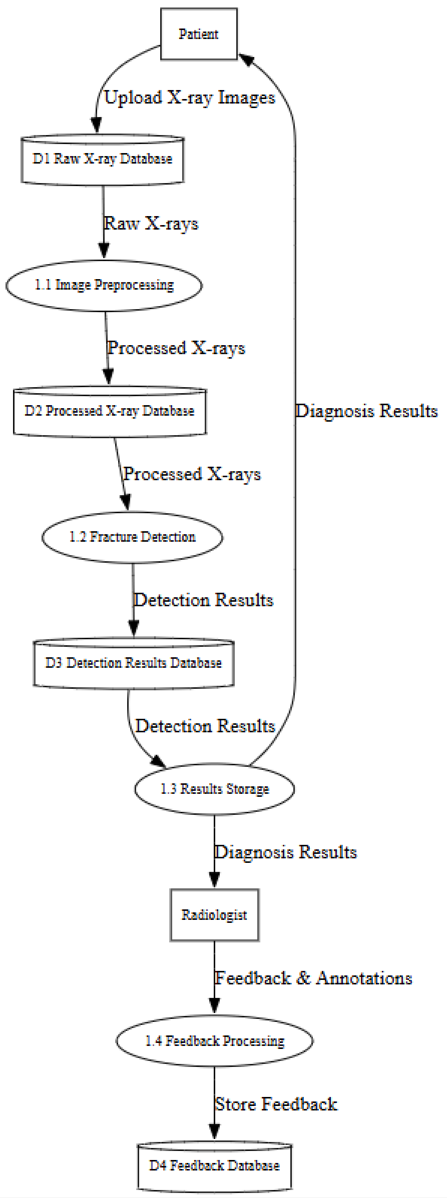

START (Month 1 - Day 1-5)
Problem Analysis (Month 1 - Day 6-10)
Analysis of Existing Systems (Month 1 - Day 11-15)
Selection & Learning Software (Month 1 - Day 16-20)
Understanding Database (Month 2 - Day 1-10)
Coding (Month 2 - Day 11-20)
Page Development (Month 2 - Day 21-30)
Testing (Month 3 - Day 1-10)
Evaluation (Month 3 - Day 11-20)
Implementation (Month 3 - Day 21-30)

Data Flow Diagram (DFD)

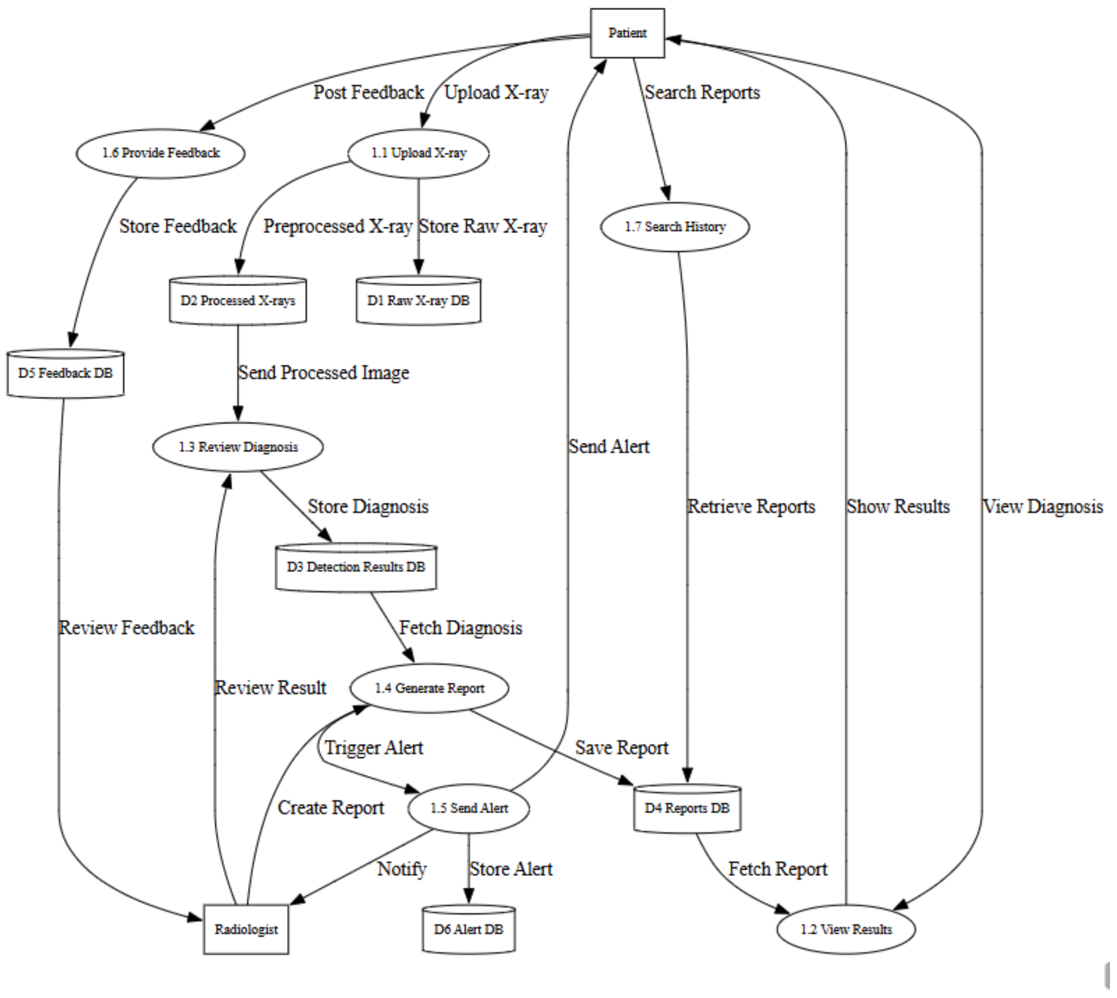
Level 0 DFD:



Level 1 DFD:


Patient
Upload X-ray Images
D1 Raw X-ray Database
Raw X-rays
1.1 Image Preprocessing
Processed X-rays
D2 Processed X-ray Database
Processed X-rays
1.2 Fracture Detection
Detection Results
D3 Detection Results Database
Detection Results
1.3 Results Storage
Diagnosis Results
Diagnosis Results
Radiologist
Feedback & Annotations
1.4 Feedback Processing
Store Feedback
D4 Feedback Database

Level 2 DFD:


Patient
Post Feedback
Upload X-ray
Search Reports
1.6 Provide Feedback
1.1 Upload X-ray
1.7 Search History
Store Feedback
Preprocessed X-ray
Store Raw X-ray
D2 Processed X-rays
D1 Raw X-ray DB
D5 Feedback DB
Send Processed Image
1.3 Review Diagnosis
Send Alert
Store Diagnosis
Retrieve Reports
Show Results
View Diagnosis
D3 Detection Results DB
Review Feedback
Fetch Diagnosis
Review Result
1.4 Generate Report
Trigger Alert
Save Report
Create Report
1.5 Send Alert
D4 Reports DB
Notify
Store Alert
Fetch Report
Radiologist
D6 Alert DB
1.2 View Results

Database Design

- **User Table:** Stores user details and uploaded images.
- **Prediction Results Table:** Stores classification outcomes and timestamps.

Input to the System

- Wrist X-ray image (PNG/JPG).
- User interaction via web interface.

Module Description

1. **Data Loading Module:** Reads X-ray paths & labels from CSVs.
2. **Preprocessing Module:** Resizes & normalizes images.
3. **Model Module:** DenseNet121 + custom classifier.
4. **Training Module:** Model training & validation.
5. **Evaluation Module:** Accuracy, precision, recall, F1-score.
6. **Prediction Module:** Real-time predictions for new X-rays.
7. **Web UI Module:**
    - Built using **Streamlit**.
    - User uploads X-ray → Model Prediction → Result displayed immediately.

Output from the System

- Classification Result: **Fractured / Normal**
- Displayed instantly on Streamlit web interface.

Testing & Security

Testing:

- Unit testing of modules.
- Validation accuracy >90%.
- Confusion matrix analysis.

Security:

- No sensitive patient data stored.
- De-identified X-ray images used.
- Secure web interface deployment.

Future Scope

- Extend to detect fractures in elbow, shoulder, etc.
- Integrate Grad-CAM visualization.
- Multi-user role-based access (Doctor/Admin).
- Mobile-friendly version of Streamlit interface.
- Integration with hospital management systems.

Bibliography

1. Densely Connected Convolutional Networks," Gao Huang et al., 2017.

2. MURA Dataset: Stanford ML Group.

3. TensorFlow & Keras Documentation.

4. Streamlit Documentation.