Title: **AgriYield Predictor**: Forecasting Crop Yield Using Environmental and Soil Data

Objective:
To develop a machine learning system that predicts crop yield based on tabular environmental and soil parameters such as rainfall, temperature, humidity, soil type, and nutrient content. The project aims to assist farmers and agricultural planners in optimizing crop production.

Outcomes:

- Understand preprocessing and feature engineering on tabular agricultural datasets.
- Learn to train and evaluate regression models for yield prediction.
- Gain experience in model performance comparison and explainability (e.g., SHAP values).
- Deliver a functional prototype with yield predictions based on user input or uploaded data.

Dataset:

- **FAO Crop Production Dataset**: Contains crop yield data from different regions.

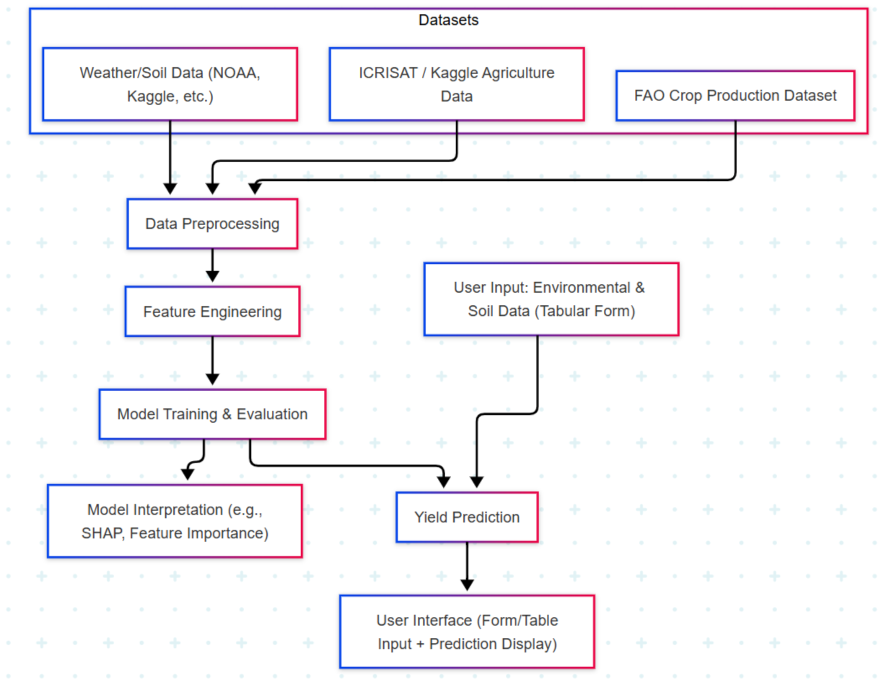
  Link: https://www.fao.org/faostat/en/#data/QCL

- **Kaggle or Indian Government Open Agriculture Data Portal**: Includes environmental and soil data by region.

  Link: https://www.kaggle.com/datasets/atharvaingle/crop-recommendation-dataset
  Link2: https://www.data.gov.in/sector/agriculture

- **Weather and Soil Data** from publicly available sources like Kaggle, NOAA, or Open Weather Map historical APIs.
  Link: https://www.kaggle.com/datasets/sobhanmoosavi/us-weather-events
  Link: https://www.ncei.noaa.gov/

Architectural diagram:


Datasets
Weather/Soil Data (NOAA, Kaggle, etc.)
ICRISAT / Kaggle Agriculture Data
FAO Crop Production Dataset
Data Preprocessing
Feature Engineering
User Input: Environmental & Soil Data (Tabular Form)
Model Training & Evaluation
Model Interpretation (e.g., SHAP, Feature Importance)
Yield Prediction
User Interface (Form/Table Input + Prediction Display)

Modules to be Implemented:

1. Data Collection & Preprocessing
2. Exploratory Data Analysis (EDA) & Feature Engineering
3. Machine Learning Model Development
4. UI for Data Input and Prediction Output
5. Evaluation and Final Deployment

Week-wise Module Implementation:

Milestone 1: Week 1 & 2 - Requirements & Dataset Preparation

- Define project scope and success metrics.
- Collect and clean crop yield, weather, and soil datasets.
- Merge datasets based on geolocation and crop type.
- Handle missing values, normalize/standardize features.

Milestone 2: Week 3 & 4 - EDA & Feature Engineering

- Visualize relationships between features and yield.
- Identify important variables (e.g., NPK levels, rainfall).
- Encode categorical features (e.g., soil type, crop name).
- Engineer new features like growing season index, heat days, etc.

Milestone 3: Week 5 & 6 - Model Development & Evaluation

- Train multiple regression models (Random Forest, XGBoost, Linear Regression).
- Evaluate using RMSE, R^2, MAE on test data.
- Use SHAP or feature importance to interpret models.
- Select best-performing model for integration.

Milestone 4: Week 7 & 8 - UI, Integration & Deployment

- Create a simple web UI for input (tabular or form).
- Show predicted yield with confidence intervals.
- Perform real-time testing and validation.
- Final documentation, deployment, and presentation.

Evaluation Criteria:

Milestone 1:

- Completion of data cleaning and preparation.
- Correct merging and processing of multiple data sources.

Milestone 2:

- Insightful EDA and logical feature engineering.
- Clear explanation of feature selection.

Milestone 3:

- Good model performance and generalization.
- Use of interpretation techniques for transparency.

Milestone 4:

- Fully functional input form or table-based UI.
- Smooth prediction flow and working backend.
- Quality documentation and a clean project presentation.

Tools & Tech Stack

To implement the AgriYield Predictor project effectively, consider the following tools and technologies:

Programming Language

- **Python**: Widely used for data analysis and machine learning tasks.

Libraries & Frameworks

- **Data Handling**: pandas, numpy
- **Visualization**: matplotlib, seaborn, plotly
- **Machine Learning**: scikit-learn, xgboost, lightgbm
- **Model Interpretation**: SHAP, eli5
- **Web Framework**: Flask or Django for developing the web interface
- **Frontend**: HTML, CSS, JavaScript, possibly with Bootstrap for responsive design

Development Tools

- **IDE**: Jupyter Notebook, VS Code
- **Version Control**: Git with platforms like GitHub or GitLab
- **Deployment**: Heroku, AWS, or Google Cloud Platform for hosting the application