Project Report: Crop Yield Prediction

1. Project Overview

Goal: To develop a machine learning model that accurately predicts crop yield (e.g., in tonnes per hectare) based on agricultural, soil, and weather data for Uttarakhand.

Team & Structure:

- **Project Manager:** All 4
- **GitHub Repository:** `Crop-Yield-Prediction-ML`
- **Team Members & Branches:**
  - 🧩 **Member 1(Jatin):** Data Collection + Cleaning (`feature/data-preprocessing`)
  - 🤖 **Member 2(Ayush):** Feature Engineering + EDA (`feature/feature-engineering`)
  - 📈 **Member 3(Rishabh) :** Model Training + Tuning (`feature/model-training`)
  - 📊 **Member 4(Aryan):** Evaluation + Visualization + Integration (`feature/evaluation`)

2. Project Execution & Workflow

This section outlines the end-to-end flow of the project, from raw data to a final model.

Phase 1: Data Collection & Cleaning (Member 1: Jatin Singh Thakur)

- **Task:** Collect all necessary raw data files. The primary dataset was `uttarakhand_crop_yield.csv` , which contained crop, year, season, area, production, rainfall, fertilizer, and pesticide data.
- **Actions:**
  - Perform initial data loading and inspection.
  - Handle basic cleaning: fix column names, check for obvious errors.
  - Perform an initial merge if data sources were separate (e.g., merging a separate "weather" file).
- **Deliverable:** A single, provisionally clean dataset (`uttarakhand_crop_yield.csv`) ready for in-depth analysis.

Phase 2: Feature Engineering & EDA (Member 2: Ayush Dobhal)

- **Task:** Perform Exploratory Data Analysis (EDA) to understand the data and create new, valuable features for the model.
- **Actions:**
  - **EDA:** Generate plots (histograms, heatmaps) to understand feature distributions and correlations.
  - **Feature Creation:** Engineer new features known to impact yield, such as:
    - `fertilizer_per_area` (Fertilizer / Area)

    - `pesticide_per_area` (Pesticide / Area)
    - `production_per_area` (Production / Area) - *This was identified as a "leaky" feature later.*
  - **Encoding:** Convert categorical features (like `Crop` and `Season`) into numeric values using one-hot encoding.
  - **Scaling:** Standardize numerical features so the model treats them equally.
- **Deliverable:** A fully-processed, model-ready file: `data/processed_dataset.csv`.

Phase 3: Model Training & Tuning (Member 3 - Rishabh Singh Nakoti)

- **Task:** Use the processed data to train, evaluate, and tune multiple machine learning models to find the single best-performing one.
- **Actions:**
  - Identify and fix critical data leakage.
  - Train and compare baseline models (Linear Regression, Random Forest, XGBoost).
  - Select the best model and tune its hyperparameters.
- **Deliverable:**
  - `models/final_model.pkl`: The final, tuned, and saved model.
  - `models/final_metrics.json`: A JSON file with the final model's performance scores. *(This phase is detailed in section 3 below).*

Phase 4: Evaluation, Visualization & Integration (Aryan)

- **Task:** Take the final model, interpret its results, visualize its performance, and create a final script demonstrating its use.
- **Actions:**
  - Verify the saved model by loading it and re-running evaluations.
  - Create data visualizations (e.g., feature importance).
  - Write a final prediction script.
- **Deliverable:** `plots/` directory with new graphs, a `predict_yield.py` script, and a final `README.md` update. *(This phase is detailed in section 4 below).*

3. Deep Dive: Rishabh - Model Training Report (Your Work)

This section details the critical work you performed. Your primary task was to take `processed_dataset.csv` and produce `final_model.pkl`.

Step 1: The Critical Discovery (Data Leakage)

The first step was to establish a baseline. However, the initial run of `model_training.py` produced "perfect" scores:

- `LinearRegression -> 'R2': 0.99997`
- `RandomForest -> 'R2': 0.9993`

This was a major red flag. A 99.99% R^2 score on a complex problem means the model isn't *predicting;* it's "cheating" by looking at the answer.

The cause was identified as Data Leakage. The features `production_per_area`, `Production`, and `Area` were leaking the target `Yield`, as `Yield` is calculated from `Production / Area`.

Step 2: The Fix & Corrected Baseline

The `model_training.py` script was immediately modified. The `load_data` function was updated to **remove the leaky columns** (`production_per_area`, `Production`, `Area`) *before* splitting the data.

Running the script again produced new, **realistic baseline scores**:

- `LinearRegression`: R^2 = 0.9846 | RMSE = 1.429
- `RandomForest`: R^2 = 0.9936 **| RMSE =** 0.916 **(Winner)**
- `XGBoost`: R^2 = 0.9904 | RMSE = 1.128

This proved that:

1. The data leakage was fixed.
2. The `RandomForest` model was the clear champion, with the highest R^2 and lowest error.

Step 3: Model Tuning

The script then automatically proceeded to "tune" the winning `RandomForest` model using `RandomizedSearchCV`. This process automatically tests many combinations of settings to find the most optimal ones.

The best parameters found were:

- `n_estimators` (number of trees): `200`
- `max_depth` (tree depth): `10`
- `min_samples_split`: `2`
- `min_samples_leaf`: `1`

Step 4: Final Model & Validation

The script saved the final, tuned model as `models/final_model.pkl`. This model was then tested one last time, yielding the **final project metrics**:

- **Final R-squared (R^2):** 0.9927
  - *(This means our model can explain 99.3% of the change in crop yield, which is an outstanding result.)*
- **Final RMSE (Error):** 0.9836
  - *(This means our model's predictions are, on average, only 0.98 units off from the actual yield.)*

Finally, `src/test_model.py` was created and run. It successfully loaded the `final_model.pkl`, re-ran the predictions, and confirmed the results, **passing the test with** ✅ `SUCCESS`.

4. The Path Forward: Plan for Aryan

Rishabh's work is complete. The following is the detailed plan for Member 4(Aryan) to complete the project.

Task 1: Understand the Model (Visualization)

The first step is to interpret *why* the model works. You need to create a new script (e.g., `src/visualize.py`). This script will:

1. Load the `models/final_model.pkl`.
2. Load the test data (using the same `load_data` and `train_test_split` logic from `src/test_model.py`).
3. Generate and save the following plots to a new `plots/` directory:
    - **Feature Importance Plot:** This is the *most important* visualization. It will show which features (like `Annual_Rainfall`, `Fertilizer`, `Crop_Wheat`, etc.) the model relies on most.
    - **Predicted vs. Actual Plot:** This is a scatter plot comparing the `y_test` (actual values) to the `y_pred` (model's predictions). It's a great visual check for accuracy.

Task 2: Final Integration (The "Application")

The final goal is to create a simple script that demonstrates how to **use the model** for a new prediction.

- Create a script named `src/predict_yield.py`.
- This script should:
    1. Load `models/final_model.pkl`.
    2. Load the *feature names* (you can get these from `X_train.columns` in your `visualize.py` script).
    3. Define a single, new "hypothetical" sample of data (e.g., a 2D array or DataFrame with one row, containing values for rainfall, fertilizer, and a specific crop).
    4. Use `model.predict()` on this new sample.
    5. Print the predicted yield in a clean, human-readable format (e.g., `Predicted Yield: 2.45 tonnes/hectare`).

Task 3: Document & Share

1. Update the main `README.md` file for the project.
2. Add the final R^2 and RMSE scores.
3. Embed the `feature_importance.png` and `predicted_vs_actual.png` plots directly in the `README.md`.
4. Merge your `feature/evaluation` branch into `main`.
5. Announce to the team that the project is complete!