Rice yield predictions using remote sensing and machine learning

15-Day Project Plan

July 14, 2025

Introduction

This document outlines a structured 15-day plan for completing a machine learning project that leverages remote sensing data to predict crop yield, specifically focusing on rice. The plan is designed to maximize efficiency by building on existing machine learning and deep learning skills while providing a rapid introduction to essential remote sensing concepts. By following this plan, the project can be completed within the tight timeframe, ensuring a high-quality outcome suitable for portfolio enhancement.

1 Phase 1: Research and Setup (Days 1–3)

1.1 Day 1: Remote Sensing Fundamentals

- Study the basics of remote sensing, including spectral bands (e.g., RGB, NIR), spatial and temporal resolution, and vegetation indices like NDVI.
- Resources: USGS Remote Sensing Basics, Sentinel-2 User Guide.
- **Tip**: Focus on optical remote sensing for this project.

1.2 Day 2: Data Sources and Tools

- Explore Sentinel-2 data via Google Earth Engine (GEE) or Copernicus Hub.
- Test the GEE Python API with a sample script to fetch imagery.
- **Tip**: Use GEE for cloud-based data access to simplify the process.

1.3 Day 3: Environment Setup

- Install necessary libraries: `rasterio`, `geopandas`, `tensorflow`, `earthengine-api`.
- Write a script to load and visualize a Sentinel-2 image.
- **Tip**: Ensure all dependencies are correctly installed to avoid delays later.

2 Phase 2: Data Collection and Preprocessing (Days 4–7)

2.1 Day 4: Data Acquisition

- Download Sentinel-2 imagery for a specific rice-growing region using GEE or Copernicus Hub.

- Filter data by date (e.g., growing season) and cloud cover (aim for less than 20%).
- **Tip**: Use GEE's built-in functions to simplify data filtering.

2.2 Day 5: Preprocessing

- Compute NDVI: `(NIR - Red) / (NIR + Red)`.
- Apply cloud masking and normalize pixel values.
- **Tip**: Use `rasterio` for handling GeoTIFF files.

2.3 Day 6: Feature Extraction

- Extract relevant features (e.g., NDVI time-series) or labels (e.g., yield proxies).
- **Tip**: Consider using vegetation indices as features for the model.

2.4 Day 7: Data Splitting

- Split the dataset into training (70%), validation (20%), and test (10%) sets.
- **Tip**: Ensure consistent coordinate systems (e.g., EPSG:4326) across datasets.

3 Phase 3: Model Development and Training (Days 8–12)

3.1 Day 8: Model Selection

- Choose a CNN architecture (e.g., 3 convolutional layers + 2 dense layers) for regression.
- **Tip**: Start with a simple model to establish a baseline.

3.2 Day 9: Implementation

- Implement the model in TensorFlow, including data loaders for efficient training.
- **Tip**: Use small image sizes (e.g., 64x64) to speed up initial training.

3.3 Day 10: Training

- Train the model using MSE loss; monitor metrics like RMSE and R^2.
- **Tip**: Use Google Colab for additional computational power if needed.

3.4 Day 11: Fine-Tuning

- Adjust hyperparameters (e.g., learning rate) and add dropout to prevent overfitting.
- **Tip**: Experiment with learning rates between 0.001 and 0.0001.

3.5 Day 12: Evaluation

- Evaluate the model on the test set and analyze performance (aim for RMSPE <10%).
- **Tip**: Visualize predictions vs. actual values to identify patterns or errors.

4 Phase 4: Deployment and Documentation (Days 13–15)

4.1 Day 13: Deployment

- Build a Streamlit app to display predictions from uploaded satellite imagery.
- **Tip**: Keep the app simple—focus on core functionality first.

4.2 Day 14: Documentation

- Write a project report covering methodology, results, and code overview.
- **Tip**: Include visualizations and metric summaries for clarity.

4.3 Day 15: Final Polish

- Debug the app, enhance visualizations, and share the project via GitHub.
- **Tip**: Log daily progress to streamline the documentation process.

Additional Notes

- **Scope**: Focus on a small, specific goal (e.g., predicting rice yield in one region).
- **Challenges**: Be mindful of cloud cover—use GEE's masking tools to mitigate issues.
- **Resources**: Refer to GEE documentation and your existing ML/DL knowledge to accelerate progress.