
OleaSat

Sensorless AI for Moroccan Olive Groves

HACKATHON EXECUTION BLUEPRINT

Precision Agriculture Innovation

2024



Table of Contents

Right-click the TOC and select "Update Field" to refresh page numbers

Table of Contents	2
Phase 1: Product Owner Perspective.....	3
The Core Concept	3
The Minimum Viable Product (MVP) Features.....	3
The Narrative for the Jury.....	3
Phase 2: Project Manager Perspective	5
The 3-Person Team Breakdown	5
Timeline (48 Hours)	5
Phase 3: AI Engineer Perspective	7
Strategy 1: The "Agronomy Formula" Approach	7
Strategy 2: The "NASA Downscaling" Hack.....	7
Strategy 3: The Transformer / Time-Series Approach	8
What You Actually Need to Build (The Stack).....	9

Phase 1: Product Owner Perspective

"The What & The Why"

As a PO, your goal is to build something that solves a real user problem, is economically viable, and has a clear user journey. Don't focus on the code yet; focus on the value.

The Core Concept

We are building a**Virtual Rain Gauge & Soil Sensors**specifically tailored for Moroccan olive trees (*Olea europaea*). It translates complex satellite and climate data into a simple SMS/WhatsApp instruction for the farmer.

The Minimum Viable Product (MVP) Features

For the hackathon, you only need to build these 3 things to prove the concept:

Feature	Description
Farm Selector	A simple map interface where the user drops a pin or draws a polygon over their olive grove (e.g., in the Meknes/Marrakech region).
Virtual Sensor Dashboard	Once selected, the UI displays the current "Water Stress Level" (Red/Yellow/Green) of that specific plot, calculated in real-time by the AI.
Actionable Alert	A generated message: "Your plot in Sidi Kacem is at 30% moisture. To maintain olive flowering, apply 15mm of water tomorrow."

Table 1: Core MVP Features

The Narrative for the Jury

"Morocco has over 1.2 million hectares of olive trees. Only large corporations can afford \$300/hectare IoT soil sensors. We are democratizing precision agriculture. Our AI uses free satellite data to give every smallholder farmer a virtual sensor in their pocket, saving up to 40% of wasted irrigation water."

The following chart illustrates the market opportunity and water savings potential:

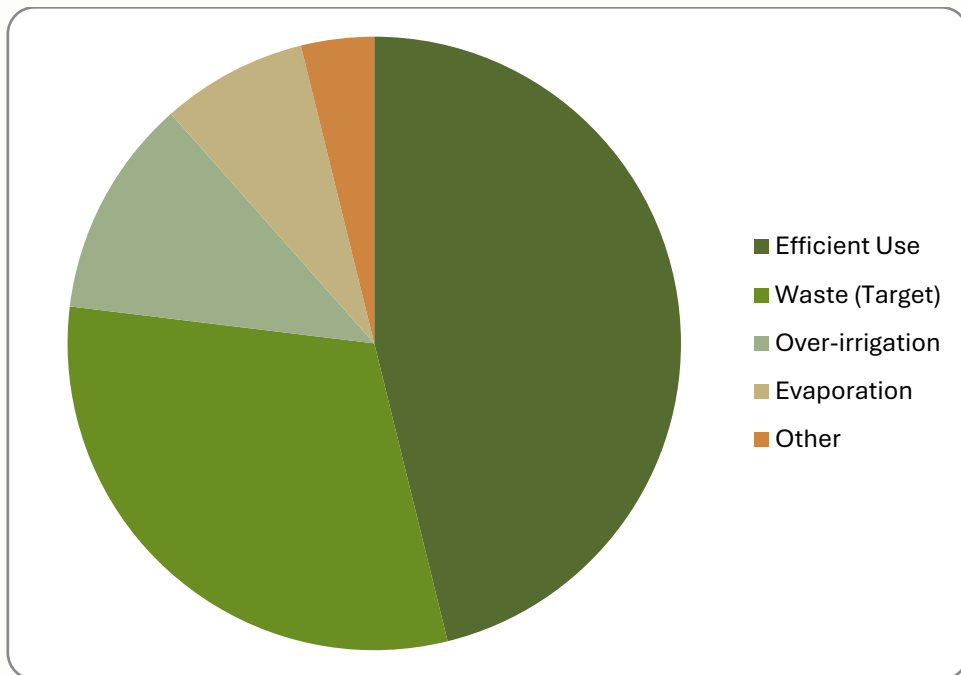


Figure 1: Water Usage Distribution in Moroccan Olive Farming

Phase 2: Project Manager Perspective

"The Execution"

A hackathon is a race against time. You need to divide and conquer. Here is how you split the work among your teammates:

The 3-Person Team Breakdown

Team Member	Responsibilities
Member 1 (Data & AI)	Focuses entirely on Google Earth Engine (GEE), fetching the data, and training the model.
Member 2 (Backend)	Builds a simple Python API (FastAPI or Flask) that takes GPS coordinates, calls Member 1's model, and returns the water recommendation.
Member 3 (Frontend)	Builds the UI (use Streamlit for extreme speed, or simple HTML/JS) and crafts the presentation slides.

Table 2: Team Roles and Responsibilities

Timeline (48 Hours)

The following bar chart shows the recommended schedule for the 48-hour hackathon:

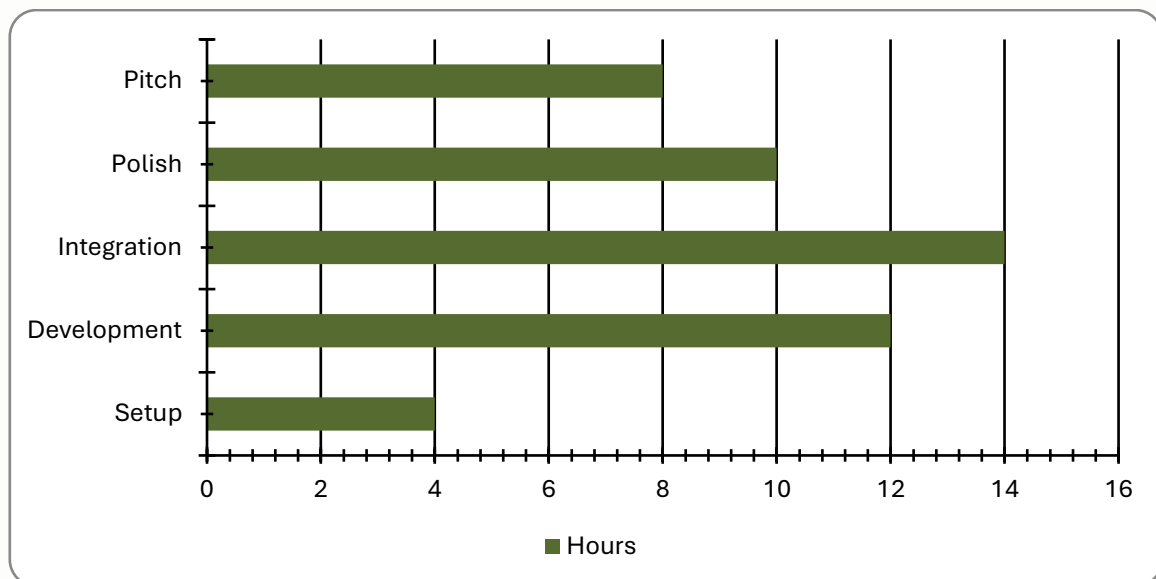


Figure 2: 48-Hour Hackathon Timeline

Hours	Activities
Hours 1-4	Finalize idea, create GitHub repo, set up GEE API access. Find a specific coordinate of a real olive farm in Morocco to use as your "demo plot".
Hours 4-16	Member 1 extracts data from GEE. Member 2 sets up the API shell. Member 3 builds the UI shell.
Hours 16-30	Train the AI model. Connect the Frontend to the Backend API.
Hours 30-40	Debugging. Making the demo look pretty. (If the AI isn't perfect, use dummy data for the presentation.)
Hours 40-48	Pitch practice, pitch practice, pitch practice.

Table 3: Detailed Timeline Breakdown

Phase 3: AI Engineer Perspective

"The Technicalities"

This is the hardest part. You mentioned: "We don't have much data, other than Sentinel or Google Earth Engine." **The Problem:** Supervised Machine Learning requires X (Features) and Y (Target/Ground Truth). You have X (Satellites), but you don't have Y (Real soil moisture data from physical sensors in Morocco).

Here are Three Strategies to solve the missing Y problem, from easiest to most advanced:

Strategy 1: The "Agronomy Formula" Approach

Easiest - No ML needed

Don't use ML. Use GEE to calculate the Water Balance.

0. **X:** Fetch Precipitation (P) and Temperature to calculate Evapotranspiration (ET0) from ERA5-Land on GEE.
1. **X:** Fetch Sentinel-2 NDVI to estimate the Crop Coefficient (Kc) of the olive trees.
2. **Output:** Crop Water Need = $ET0 \times Kc$. If Precipitation < Crop Water Need, tell the farmer to irrigate the difference.

Pros: Guaranteed to work. Agronomically sound.

Cons: Lacks the "AI/Deep Learning" wow factor.

Strategy 2: The "NASA Downscaling" Hack

Best for Hackathons - High Wow Factor

Use a low-resolution satellite that does measure soil moisture as your Y, and use high-resolution satellites as your X.

3. **Target (Y):** NASA SMAP satellite measures actual soil moisture (available on GEE), but resolution is 9km x 9km - useless for small farms.
4. **Features (X):** Sentinel-1 (Radar backscatter) and Sentinel-2 (Optical/NDVI) with 10m x 10m resolution.
5. **Model:** Train a model (Random Forest or XGBoost) to predict the 9km SMAP value using the 10m Sentinel data.

"We built an AI that super-resolves NASA soil moisture data. We take 9km data and make it 10m farm-level data using Sentinel-1 radar fusion." (Judges will love this)

Strategy 3: The Transformer / Time-Series Approach

Hardest

If you strictly want to use deep learning (LSTMs or Transformers):

6. **Extract Time Series:** Use GEE to extract the last 3 years of daily data for 1,000 random olive grove coordinates in Morocco.
7. **Features (X):** Sequence of [Precipitation, Temp, NDWI (Water Index), NDVI] for Days T-30 to T-1.
8. **Target (Y):** NDWI (Normalized Difference Water Index) at Day T. NDWI is an optical proxy for plant water stress.
9. **Architecture:** Pass the 30-day sequence into a Temporal Fusion Transformer (TFT) or an LSTM.
10. **Output:** The model predicts if the olive tree canopy will show water stress tomorrow based on weather and satellite history.

What You Actually Need to Build (The Stack)

Component	Description
Google Earth Engine	Python API (geemap). Script takes [longitude, latitude] and date_range. Returns Pandas DataFrame with: NDVI, NDWI, VV/VH (Sentinel-1), Temperature, Precipitation.
The Model	Scikit-learn (Random Forest Regressor) for Strategy 2, or PyTorch for Strategy 3. Save as .pkl or .pt file.
The API	Simple FastAPI script. One endpoint: /predict?lat=X&lon=Y. Queries GEE for last 10 days, runs model, returns {"recommended_irrigation_liters": 20}.
The UI	Use Streamlit. Build interactive map in 10 lines with streamlit-folium. User clicks map, Streamlit calls FastAPI, displays result.

Table 4: Technical Stack Components

Secret Hackathon Tip for the Pitch

Don't try to process satellite data for the whole country in real-time during the demo. GEE will time out or rate-limit you, and the demo will fail.

Pre-calculate the data for 3 specific olive farms (e.g., one healthy, one dry, one dying). Hardcode these coordinates in your demo. When presenting, click on those specific farms to instantly show the AI prediction. Tell the judges: "For this MVP, we have pre-processed these regions to ensure a smooth demo, but the architecture scales globally via Earth Engine."

OleaSat

Democratizing Precision Agriculture

Google Earth Engine · Sentinel · AI/ML

© 2024 OleaSat Project