Patrick Ndowa

Final Report.

ISOM 835 Final Project

OPTION 1 — Optimization Track

Smart Farm Planner: Crop Mix Optimization for Small Farmers

A real prescriptive analytics product with real-world impact.

Problem

Small farmers don't know the optimal crop mix for maximum profit under:

- land constraints
- water availability
- fertilizer cost
- seasonal price volatility.

Solution

A Gurobi-powered optimizer that decides the **optimal crop portfolio**.

Inputs

- Available land (acres)
- Maximum water
- Budget
- Crop prices (auto-loaded dataset)
- Yield per crop
- Labor needed
- Risk tolerance level (adds variance penalty term)

Outputs

- Optimal crop mix (acres per crop)
- Expected profit
- Resource utilization

- Sensitivity analysis
- Recommendation narrative (“Plant 40% maize, 35% soybeans…”)

Why this is strong

- Fits my background as a **farmer** → authentic and deep
- Uses Gurobi → employer-facing wow factor
- Deployable with Streamlit
- Can be extended into a startup idea
- Extremely portfolio-ready

1. Problem statement (elevator pitch)

Smallholder farmers must decide how to allocate limited land and water across available crops to maximize profit while respecting resource limits and controlling risk from price/yield uncertainty. Smart Farm Planner is a prescriptive optimization app (Gurobi + Streamlit) that recommends the optimal crop mix (acres per crop), expected profit, resource usage, and simple sensitivity insights.

---

2. Product Requirements Document (PRD)

Users: Smallholder farmers, farm cooperatives, ag extension agents, ag-tech recruiters/interviewers.

Core MVP features:

- Inputs: total land (acres), water budget, overall budget for inputs, labor available, risk aversion slider, available crop list (name, expected yield, water per acre, input cost per acre, min/max acres), price scenarios (or use default historical averages).
- Engine: Gurobi-based optimizer (QP if including variance penalty).
- Outputs: acres per crop, expected profit, utilization of water/labor/budget, binding constraints, sensitivity (shadow prices), short plain-English recommendation and "why".
- Extras: CSV export of recommendations, charts (pie of land allocation, bar of expected profit by crop), scenario comparison (low/med/high price).

Nonfunctional:

- Deployable as Streamlit app
- Clean README, requirements.txt
- Well-documented code for interviews

---

3. Data needs (schema / sample)

Crop table (CSV)

- crop (str)
- min_acres (float)
- max_acres (float)
- yield_per_acre (float) — expected yield (units/acre)
- input_cost_per_acre (float) — seed, fertilizer, labor, USD/acre
- water_per_acre (float) — cubic meters per acre (or normalized units)
- labor_per_acre (float) — labor hours per acre
- price_mean (float) — expected price per unit (USD/unit)
- price_low (float), price_high (float) — scenario bounds (optional)

Farm-level inputs (UI fields or JSON)

- total_acres (float)
- water_budget (float)
- labor_budget (float)
- input_budget (float)
- risk_aversion (0-1 slider) — higher => more conservative

Optional historical price series per crop for scenario sampling.

---

4. Optimization formulation (math)

Decision variables:

- xi≥0xi≥0 = acres to plant of crop ii

Parameters:

- pipi = expected price for crop ii (USD/unit)
- yiyi = expected yield per acre for crop ii (units/acre)
- cici = input cost per acre for crop ii (USD/acre)
- wiwi = water use per acre for crop ii
- lili = labor per acre for crop ii
- AA = total available acres
- WW = total water budget
- LL = total labor budget
- BB = budget for inputs
- ximin,ximaxximin,ximax = min/max acres allowed per crop

- Risk term: we'll use scenario-based variance of profit. Suppose we have S price scenarios $p_{i,s}$. Profit under scenario s:
  $\Pi_s=\sum_i x_i(p_{i,s}y_i-c_i)$
  Let $\bar{\Pi}$ be expected profit across scenarios. Variance (or second moment) can be approximated and a penalty $\lambda\cdot \mathrm{Var}(\Pi)$ added to objective.

Objective (maximize risk-adjusted expected profit):

$$\max_x \bar{\Pi}-\lambda\cdot \mathrm{Var}(\Pi)$$

Where $\bar{\Pi}=\frac{1}{S}\sum_s\sum_i x_i(p_{i,s}y_i-c_i)$ and $\mathrm{Var}(\Pi)=\frac{1}{S}\sum_s(\Pi_s-\bar{\Pi})^2$

Constraints:

1. Land: $\sum_i x_i\le A$
2. Water: $\sum_i w_i x_i\le W$
3. Labor: $\sum_i l_i x_i\le L$
4. Input budget: $\sum_i c_i x_i\le B$
5. Bounds: $x_i^{min}\le x_i\le x_i^{max}$
6. Nonnegativity.

Notes:

- If you set $\lambda=0$ it's a linear LP. If $\lambda>0$ and you implement variance penalty, objective becomes quadratic (Gurobi handles QP).
- You may also add crop rotation constraints later (binary variables) — but MVP should stay continuous for speed.

---

5. Implementation: Gurobi & Streamlit starter

Below I provide runnable Python code for:

- generating synthetic data
- solving linear risk-neutral LP (expected profit)
- solving risk-aware QP (variance penalty)
- a Streamlit app skeleton that calls the solver and displays results

```
# optimizer.py
import numpy as np
import pandas as pd
from gurobipy import Model, GRB, QuadExpr


def generate_synthetic_crops():
  # Example crops
  crops = [
    {'crop': 'Maize', 'yield_per_acre': 1500, 'input_cost_per_acre': 200, 'water_per_acre': 100, 'labor_per_acre': 10, 'price_mean': 0.15, 'min_acres': 0, 'max_acres': 100},
    {'crop': 'Soybean', 'yield_per_acre': 1200, 'input_cost_per_acre': 180, 'water_per_acre': 80, 'labor_per_acre': 8, 'price_mean': 0.18, 'min_acres': 0, 'max_acres': 100},
    {'crop': 'Groundnut', 'yield_per_acre': 900, 'input_cost_per_acre': 150, 'water_per_acre': 60, 'labor_per_acre': 12, 'price_mean': 0.25, 'min_acres': 0, 'max_acres': 80},
    {'crop': 'Sorghum', 'yield_per_acre': 1100, 'input_cost_per_acre': 160, 'water_per_acre': 70, 'labor_per_acre': 7, 'price_mean': 0.13, 'min_acres': 0, 'max_acres': 100},
  ]
  return pd.DataFrame(crops)


def sample_price_scenarios(df, S=50, vol=0.15, seed=42):
  rng = np.random.default_rng(seed)
  prices = {}
  for i, row in df.iterrows():
    mu = row['price_mean']
    # log-normal-like sampling but keep positive
    scenario_prices = rng.normal(mu, mu*vol, size=S)
    scenario_prices = np.clip(scenario_prices, a_min=mu*0.5, a_max=mu*1.5)
```

```
        prices[row['crop']] = scenario_prices
    # DataFrame S x crops
    return pd.DataFrame(prices)


def solve_optimizer(df_crops, price_scenarios,
                    A=100, W=8000, L=900, B=15000, risk_aversion=0.0):
    crops = df_crops['crop'].tolist()
    y = df_crops.set_index('crop')['yield_per_acre'].to_dict()
    c = df_crops.set_index('crop')['input_cost_per_acre'].to_dict()
    w = df_crops.set_index('crop')['water_per_acre'].to_dict()
    l = df_crops.set_index('crop')['labor_per_acre'].to_dict()
    price_mean = df_crops.set_index('crop')['price_mean'].to_dict()

    # Precompute per-acre profit under each scenario
    scenarios = price_scenarios.values  # S x n_crops
    S = scenarios.shape[0]
    crop_index = {crop:i for i,crop in enumerate(crops)}
    per_acre_profit_s = {}
    for s in range(S):
        for crop in crops:
            p = price_scenarios.iloc[s][crop]
            per_acre_profit_s.setdefault(s, {})[crop] = p * y[crop] - c[crop]

    # Expected per-acre profit
    exp_profit_per_acre = {crop: np.mean(price_scenarios[crop].values) * y[crop] - c[crop] for
crop in crops}
```

```
# Create gurobi model
m = Model('farm_opt')
m.Params.OutputFlag = 0  # silent
x = {crop: m.addVar(lb=df_crops.loc[df_crops['crop']==crop,'min_acres'].values[0],
            ub=df_crops.loc[df_crops['crop']==crop,'max_acres'].values[0],
            name=f"x_{crop}")
   for crop in crops}

# Land
m.addConstr(sum(x[c] for c in crops) <= A, name='land')
# Water
m.addConstr(sum(w[c]*x[c] for c in crops) <= W, name='water')
# Labor
m.addConstr(sum(l[c]*x[c] for c in crops) <= L, name='labor')
# Input budget
m.addConstr(sum(c[crop]*x[crop] for crop in crops) <= B, name='budget')

# Objective: expected profit minus risk penalty * variance
# Expected profit (linear)
exp_profit_expr = sum(exp_profit_per_acre[crop]*x[crop] for crop in crops)

if risk_aversion <= 0:
  m.setObjective(exp_profit_expr, GRB.MAXIMIZE)
else:
  # Build variance term: Var = (1/S)*sum_s (Pi_s - mean)^2
```

```
    # Pi_s = sum_i per_acre_profit_s[s][i] * x_i
    # mean = exp_profit_expr
    # Var = (1/S) * sum_s (sum_i (a_{i,s} x_i) - sum_i (a_i_bar x_i))^2
    # Expand: quadratic in x. We'll build QuadExpr
    quad = QuadExpr()
    a_bar = {crop: exp_profit_per_acre[crop] for crop in crops}
    for s in range(S):
        # coefficients for scenario s: a_{i,s}
        a_s = {crop: per_acre_profit_s[s][crop] for crop in crops}
        # build (sum_i (a_{i,s} - a_bar_i) * x_i)^2
        for i in crops:
            for j in crops:
                coef = (a_s[i] - a_bar[i]) * (a_s[j] - a_bar[j]) / S
                if coef != 0:
                    quad.addTerms(coef, x[i], x[j])
    # Objective: maximize exp_profit_expr - risk_aversion * quad
    obj = exp_profit_expr - risk_aversion * quad
    m.setObjective(obj, GRB.MAXIMIZE)

m.optimize()

result = {crop: x[crop].X for crop in crops}
# compute expected profit and scenario profits
expected_profit = sum(exp_profit_per_acre[crop]*result[crop] for crop in crops)
scenario_profits = []
for s in range(S):
```

```
        Pi_s = sum(per_acre_profit_s[s][crop]*result[crop] for crop in crops)
        scenario_profits.append(Pi_s)
    var_profit = np.var(scenario_profits)
    return {
        'x': result,
        'expected_profit': expected_profit,
        'var_profit': var_profit,
        'scenario_profits': scenario_profits,
        'model': m
    }


if __name__ == "__main__":
    df = generate_synthetic_crops()
    price_scen = sample_price_scenarios(df, S=100, vol=0.2)
    res = solve_optimizer(df, price_scen, A=120, W=9000, L=1200, B=20000,
risk_aversion=0.01)
    print("Acres allocation:")
    for k,v in res['x'].items():
        print(k, round(v,2))
    print("Expected profit:", round(res['expected_profit'],2))
    print("Profit variance:", round(res['var_profit'],2))
```

I acknowledge the use of Artificial Intelligence to enhance my thoughts and expand my ideas with generations of codes which I had to develop and finalise with my human intuition