Smarter Yields

Building a Crop Recommendation Model Using Machine Learning

The Agriculturol Chollenge

"What should I plant?" is the most critical question for a farmer. How can we use data to provide a confident answer?

The Solution: AgriTech in Action

What is a Crop Model?

It is an intelligent system that processes environmental and soil data to suggest the most suitable crop for a specific piece of land.

- Moves farmers from "tradition" to "data-driven decisions."
- Aims to maximize yield, sustainability, and profitability.
- A key tool in modern Precision Agriculture.

How the Model Works

Data Collection

Gathering historical data on soil composition, climate patterns, and geographic factors.

Model Training

Using ML algorithms (like Random Forest) to find hidden patterns in the data.

Recommendation

The trained model predicts the optimal crop for a new, unseen piece of land.

The Fuel: Key Data Inputs

Soil & Land Data

- Nitrogen (N) Level
- Phosphorus (P) Level
- Potassium (K) Level
- Soil pH Value
- Soil Type

Environmental Data

- Average Temperature
- Average Humidity
- Annual Rainfall (in mm)
- Altitude/ Elevation

Data Profile: Optimal Nutrient Needs

Crop	Nitrogen (N)	Phosphorus (P)	Potassium (K)	Optimal pH
Rice	80-100	40-60	40-60	5.5–B5
Maize	100-120	50-70	60-80	6.0 - 7.0
Cotton	110-130	50-60	50-60	'i8-S5
Jute	60-80	3050	30-50	6.0 - 7.5

Data Profile: OptimolRoinfoll

The model learns these optimal ranges. Rice, for example, requires significantly more water than Cotton.

The "Broin^. Choosing the Algorithm

Random Forest: High Accuracy. An ensemble of "Decision Trees." Often the best-performing model for this task as it captures complex, non-linear interactions.

Decision Tree: Easy to Interpret.AsimpIeflowchart of"if-then" rules based on the data (e.g.,"IF rainfall > 1000mm AND N >80...").

Gaussian NaTve Bayes: Fast &Efficient. A probabilistic model that calculates the likelihood a crop is suitable based on the inputs.

@ Support Vector Machine (SVM): Powerful. Finds the optimal "boundary" in the data that separates the different crop classes.

Why Random Forest Often Wins

The Wisdom of the Crowd

Instead of relying on one "Decision Tree," a Random Forest builds "hundreds* of them, each with a slightly different perspective. It then takes the majority vote. This "ensemble" approach makes it highly accurate and prevents errors from "overfitting" to one specific dataset.

Beyond the Basics: Advanced Factors

Economic Viability

A more advanced model can be enhanced to also consider:

- Market Prices
- Transportation Costs
- Profitability Analysis

Logistical Factors

Further enhancements can include:

- Resource Availability(Labor,Machinery)
- Crop Rotation Schedules
- Local & Global Demand

The Impact of Smart Farming

This isn't just a model; it's a tool for sustainable and profitable agriculture.

- Increases crop yield & quality
- Reduces resource waste(fertilizer,water)
- Boosts farmer profitability
- Supports regional food security

```python
import tkinter as tk
from tkinter import ttk, messagebox
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import random

# ------------------------------
# Synthetic data + trained model
# ------------------------------
np.random.seed(42)
n_samples = 300
data = {
    "Nitrogen": np.random.randint(0, 150, n_samples),
    "Phosphorus": np.random.randint(0, 150, n_samples),
    "Potassium": np.random.randint(0, 150, n_samples),
    "Temperature": np.random.uniform(10, 40, n_samples),
    "Humidity": np.random.uniform(20, 90, n_samples),
    "pH": np.random.uniform(4, 9, n_samples),
    "Rainfall": np.random.uniform(50, 300, n_samples),
    "Crop": np.random.choice(
        ["Rice", "Wheat", "Maize", "Cotton", "Sugarcane", "Barley", "Pulses", "Coffee"],
        n_samples
    ),
}
df = pd.DataFrame(data)
le = LabelEncoder()
df["CropLabel"] = le.fit_transform(df["Crop"])
X = df.drop(columns=["Crop", "CropLabel"])
y = df["CropLabel"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

```
BASE_INFO = {
    "Rice": {"Cultivation": "Paddy fields; transplant 20–30 day seedlings; maintain levees.",
             "Water": "High — standing water required for much of the season.",
             "Pests": "Stem borer, leaf folder; watch seedling & tillering stages.",
             "Harvest": "When grains are golden and moisture ~20%.",
             "Fertilizer": "Split N application; use urea and basal P/K.",
             "Soil": "Clayey loam or alluvial soils (good water retention).",
             "Climate": "Warm, humid climate; heavy rainfall preferred."},
    "Wheat": {"Cultivation": "Sow in well-drained loamy soil; use certified seeds; maintain row spacing.",
              "Water": "Moderate — critical during tillering & grain filling.",
              "Pests": "Rusts, aphids; monitor during cool, humid weather.",
              "Harvest": "Harvest when straw turns yellow and grains hard.",
              "Fertilizer": "Basal N and top-dress at tillering; balanced NPK.",
              "Soil": "Loamy soils with good drainage.",
              "Climate": "Cool to moderate temperatures; less humid than rice."},
    "Maize": {"Cultivation": "Sow 3–5 cm deep; ensure fertile soil; plant in rows for mechanized weeding.",
              "Water": "Higher need at flowering & grain filling stages.",
              "Pests": "Stem borers, cutworms; use crop rotation.",
              "Harvest": "Harvest when cobs dry and kernels hard.",
              "Fertilizer": "Apply N heavily early; add P & K as needed.",
              "Soil": "Well-drained loam with organic matter.",
              "Climate": "Warm climate; frost-free period needed."},
    "Cotton": {"Cultivation": "Plant stem cuttings; prefer sunny, well-drained fields.",
               "Water": "Moderate — avoid waterlogging; water more during boll formation.",
               "Pests": "Bollworm, jassids; use integrated pest management.",
               "Harvest": "Harvest when bolls open and lint is fluffy.",
               "Fertilizer": "Balanced NPK; potassium important for lint quality.",
               "Soil": "Black cotton soil or deep loamy soils.",
               "Climate": "Warm climate; long growing season."},
    "Sugarcane": {"Cultivation": "Plant setts in rows; incorporate green manure.",
                  "Water": "High — regular irrigation required.",
                  "Pests": "Borers, aphids; maintain field sanitation.",
                  "Harvest": "10–18 months depending on variety; harvest when cane is mature.",
                  "Fertilizer": "High N and K requirements; split applications.",
                  "Soil": "Sandy loam to clay loam with organic matter.",
                  "Climate": "Tropical to subtropical climates best."},
    "Barley": {"Cultivation": "Sow in cooler season; good seedbed preparation.",
```

```
                "Water": "Moderate — avoid waterlogging.",
                "Pests": "Rusts; keep disease-resistant varieties when available.",
                "Harvest": "When spikes yellow and grains firm.",
                "Fertilizer": "Balanced NPK; lower N than wheat.",
                "Soil": "Sandy loam to loam.",
                "Climate": "Cool to moderate; tolerant to poorer soils."},
    "Pulses": {"Cultivation": "Sow after rains; seeds often inoculated with Rhizobium.",
                "Water": "Low to moderate — many pulses are drought tolerant.",
                "Pests": "Pod borer and aphids; timely monitoring helps.",
                "Harvest": "Harvest before pods shatter; dry properly.",
                "Fertilizer": "Low N (fix nitrogen), apply P & K as needed.",
                "Soil": "Light to medium soils; good drainage.",
                "Climate": "Often tolerant to semi-arid conditions."},
    "Coffee": {"Cultivation": "Shade-grown in rich, well-drained soils; careful pruning.",
                "Water": "Regular moisture but good drainage required.",
                "Pests": "Berry borer, leaf rust; pruning and sanitation important.",
                "Harvest": "Pick ripe red cherries by hand.",
                "Fertilizer": "Apply compost and balanced NPK; micronutrients helpful.",
                "Soil": "Rich, slightly acidic soils (pH ~6–6.5).",
                "Climate": "Tropical highlands; cooler nights and shade preferred."}
}

EXTRA_TIPS = [
    "Rotate crops to keep soil healthy.",
    "Use mulching to retain soil moisture and suppress weeds.",
    "Prefer certified seeds for better germination and yields.",
    "Maintain good drainage to avoid root diseases.",
    "Use organic manure to improve soil structure."
]


# ----------------------------
# App Class
# ----------------------------
class CropRecommendationApp:
    def __init__(self, root):
        self.root = root
```

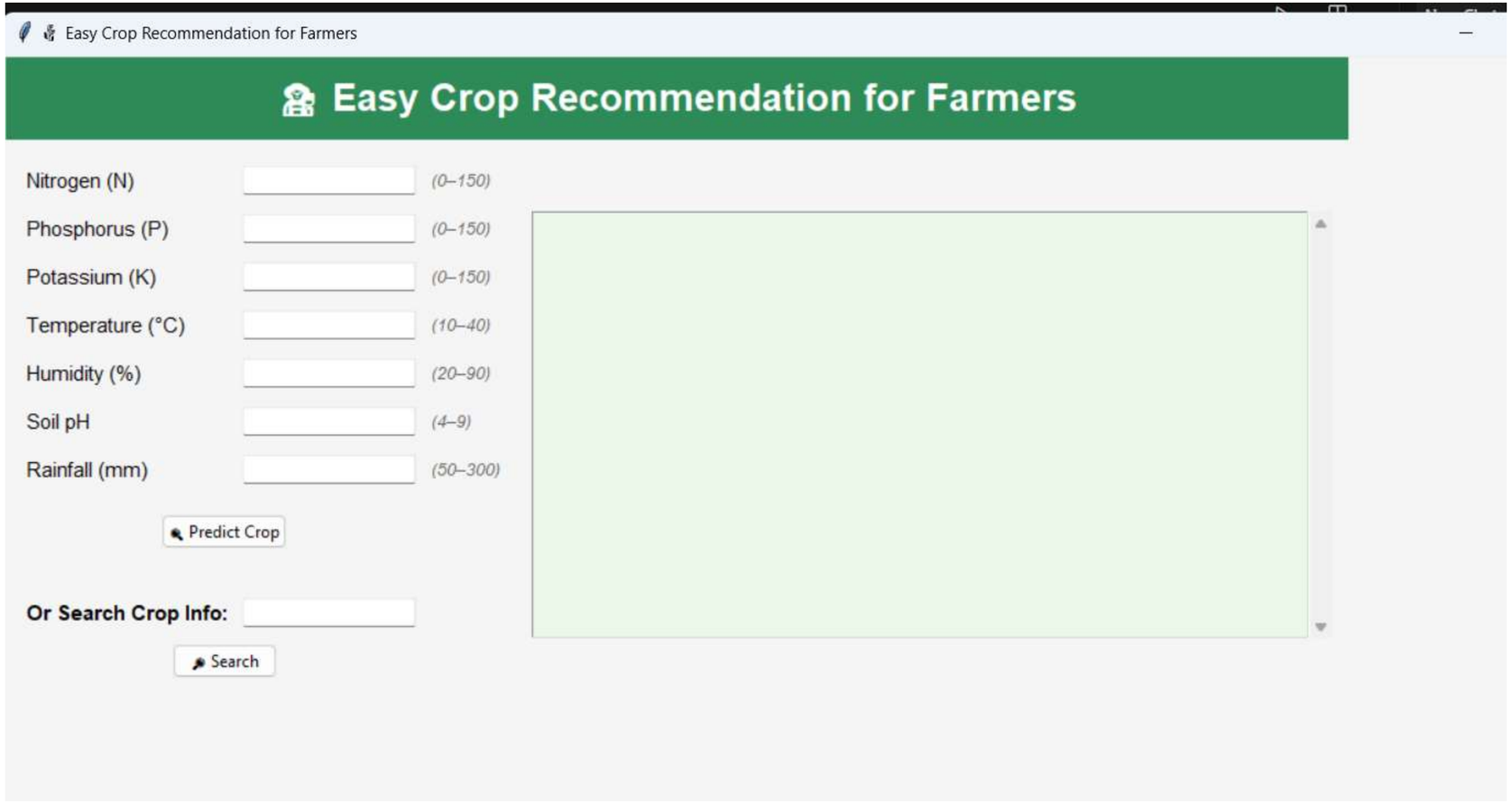
```python
class CropRecommendationApp:
    def __init__(self, root):
        self.root = root
        self.root.title("🌾 Easy Crop Recommendation for Farmers")
        self.root.geometry("1180x780")

        # Scrollable main window
        self.main_canvas = tk.Canvas(root, bg="#f5f5f5")
        self.v_scroll = ttk.Scrollbar(root, orient="vertical", command=self.main_canvas.yview)
        self.main_canvas.configure(yscrollcommand=self.v_scroll.set)
        self.v_scroll.pack(side="right", fill="y")
        self.main_canvas.pack(side="left", fill="both", expand=True)

        self.container = tk.Frame(self.main_canvas, bg="#f5f5f5")
        self.main_canvas.create_window((0, 0), window=self.container, anchor="nw")
        self.container.bind("<Configure>", lambda e: self.main_canvas.configure(scrollregion=self.main_canvas.bbox("all")))

        title = tk.Frame(self.container, bg="#2e8b57")
        title.pack(fill="x")
        tk.Label(title, text="👨‍🌾 Easy Crop Recommendation for Farmers", font=("Helvetica", 20, "bold"),
                 bg="#2e8b57", fg="white", pady=12).pack()

        body = tk.Frame(self.container, bg="#f5f5f5", padx=12, pady=12)
        body.pack(fill="both", expand=True)

        left_col = tk.Frame(body, bg="#f5f5f5")
        left_col.grid(row=0, column=0, sticky="n")
        right_col = tk.Frame(body, bg="#f5f5f5")
        right_col.grid(row=0, column=1, sticky="n", padx=(20,0))

        # Input form
        self.entries = {}
        fields = [
            ("Nitrogen (N)", "Nitrogen", "(0–150)"),
            ("Phosphorus (P)", "Phosphorus", "(0–150)"),
            ("Potassium (K)", "Potassium", "(0–150)"),
            ("Temperature (°C)", "Temperature", "(10–40)"),
```

```python
class CropRecommendationApp:
    def __init__(self, root):
            ("Soil pH", "pH", "(4–9)"),
            ("Rainfall (mm)", "Rainfall", "(50–300)")
        ]
        for i, (lbl, key, note) in enumerate[tuple[str, str, str]](fields):
            tk.Label(left_col, text=lbl, font=("Arial", 11), bg="#f5f5f5").grid(row=i, column=0, sticky="w", pady=6)
            ent = ttk.Entry(left_col, width=20)
            ent.grid(row=i, column=1, padx=8)
            tk.Label(left_col, text=note, font=("Arial", 9, "italic"), fg="#666", bg="#f5f5f5").grid(row=i, column=2, sticky="W")
            self.entries[key] = ent

        ttk.Button(left_col, text="🔍 Predict Crop", command=self.predict_crop).grid(row=len(fields), column=0, columnspan=2, pady=15)

        # Search bar
        tk.Label(left_col, text="Or Search Crop Info:", bg="#f5f5f5", font=("Arial", 11, "bold")).grid(row=len(fields)+1, column=0, sticky="w
        self.search_var = tk.StringVar()
        self.search_entry = ttk.Entry(left_col, textvariable=self.search_var, width=20)
        self.search_entry.grid(row=len(fields)+1, column=1, pady=(20,5))
        ttk.Button(left_col, text="🔎 Search", command=self.search_crop).grid(row=len(fields)+2, column=0, columnspan=2, pady=5)

        # Right side info area
        self.result_label = tk.Label(right_col, text="", font=("Helvetica", 14, "bold"), bg="#f5f5f5", fg="#2e8b57")
        self.result_label.pack(anchor="nw", pady=(6,6))

        info_outer = tk.Frame(right_col)
        info_outer.pack(fill="both", expand=False)
        self.info_text = tk.Text(info_outer, width=70, height=18, wrap="word", bg="#eaf9ea", font=("Arial", 11))
        info_scroll = ttk.Scrollbar(info_outer, orient="vertical", command=self.info_text.yview)
        self.info_text['yscrollcommand'] = info_scroll.set
        self.info_text.pack(side="left", fill="both", expand=True)
        info_scroll.pack(side="right", fill="y")
        self.info_text.config(state="disabled")

        self.chart_frame = tk.Frame(right_col, bg="#f5f5f5", pady=8)
        self.chart_frame.pack(fill="both", expand=False)
```

```python
class CropRecommendationApp:
        self.main_canvas.bind_all("<MouseWheel>", self._on_mousewheel)

    def _on_mousewheel(self, event):
        self.main_canvas.yview_scroll(int(-1*(event.delta/120)), "units")

    def predict_crop(self):
        try:
            inputs = {k: float(v.get()) for k, v in self.entries.items()}
        except ValueError:
            messagebox.showerror("Invalid Input", "Please enter numeric values for all fields.")
            return

        # Validation ranges
        ranges = {"Nitrogen": (0,150), "Phosphorus": (0,150), "Potassium": (0,150),
                  "Temperature": (10,40), "Humidity": (20,90), "pH": (4,9), "Rainfall": (50,300)}

        for k, (low, high) in ranges.items():
            if not (low <= inputs[k] <= high):
                messagebox.showerror("Invalid Input", f"{k} must be between {low} and {high}.")
                return

        arr = np.array(list[float](inputs.values())).reshape(1, -1)
        pred_label = model.predict(arr)[0]
        pred_crop = le.inverse_transform([pred_label])[0]
        probs = model.predict_proba(arr)[0]
        confidence = np.max(probs) * 100

        self.result_label.config(text=f"🌿 Recommended Crop: {pred_crop}     ✅ Confidence: {confidence:.2f}%")
        self._show_crop_info(pred_crop, inputs)
        self._show_chart(probs)
        self.container.update_idletasks()

    def search_crop(self):
        crop = self.search_var.get().capitalize().strip()
        if crop not in BASE_INFO:
            messagebox.showerror("Not Found", f"Crop '{crop}' not found.")
```

```python
class CropRecommendationApp:
    def search_crop(self):
        self._show_crop_info(crop, {})

    def _show_crop_info(self, crop, inputs):
        base = BASE_INFO.get(crop, {})
        final_text = f"🌾 {crop} Cultivation Guide\n" + "-"*56 + "\n\n"
        for heading, text in base.items():
            final_text += f"🔹 {heading}: {text}\n\n"
        final_text += "📝 Extra Tips:\n"
        for t in random.sample(EXTRA_TIPS, k=2):
            final_text += f"   • {t}\n"

        self.info_text.config(state="normal")
        self.info_text.delete(1.0, tk.END)
        self.info_text.insert(tk.END, final_text)
        self.info_text.config(state="disabled")

    def _show_chart(self, probs):
        for w in self.chart_frame.winfo_children():
            w.destroy()
        crops = le.classes_
        fig, ax = plt.subplots(figsize=(5.8, 3.2))
        ax.barh(crops, probs, color="#6aa84f")
        ax.set_xlabel("Prediction Probability")
        ax.set_title("Model Confidence by Crop")
        plt.tight_layout()
        canvas = FigureCanvasTkAgg(fig, master=self.chart_frame)
        widget = canvas.get_tk_widget()
        widget.pack()
        canvas.draw()
        plt.close(fig)

# ------------------------------
# Run the App
# ------------------------------
```

```python
# ----------------------------
# Run the App
# ----------------------------
if __name__ == "__main__":
    root = tk.Tk()
    app = CropRecommendationApp(root)
    root.mainloop()
```


Easy Crop Recommendation for Farmers
Easy Crop Recommendation for Farmers
Nitrogen (N)
(0–150)
Phosphorus (P)
(0–150)
Potassium (K)
(0–150)
Temperature (°C)
(10–40)
Humidity (%)
(20–90)
Soil pH
(4–9)
Rainfall (mm)
(50–300)
Predict Crop
Or Search Crop Info:
Search

Easy Crop Recommendation for Farmers

Nitrogen (N)	30	(0–150)
Phosphorus (P)	50	(0–150)
Potassium (K)	78	(0–150)
Temperature (°C)	20	(10–40)
Humidity (%)	34	(20–90)
Soil pH	5	(4–9)
Rainfall (mm)	300	(50–300)

Predict Crop

Or Search Crop Info:

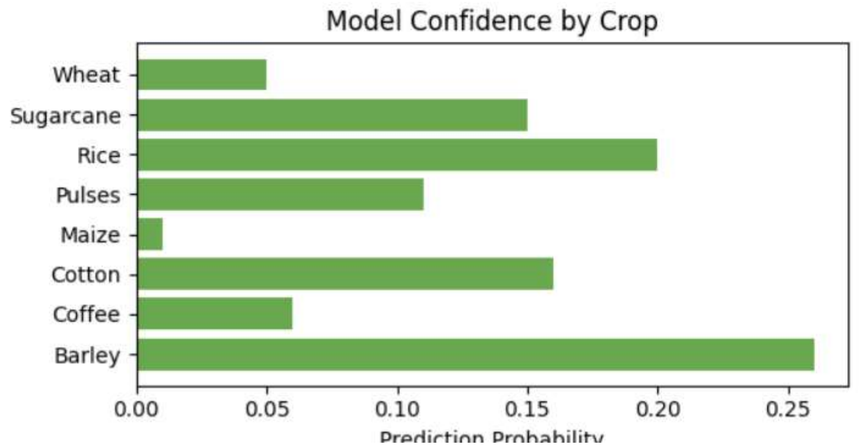
Search

Recommended Crop: Barley **Confidence: 26.00%**

Barley Cultivation Guide

- Cultivation: Sow in cooler season; good seedbed preparation.
- Water: Moderate — avoid waterlogging.
- Pests: Rusts; keep disease-resistant varieties when available.
- Harvest: When spikes yellow and grains firm.
- Fertilizer: Balanced NPK; lower N than wheat.
- Soil: Sandy loam to loam.
- Climate: Cool to moderate; tolerant to poorer soils.

Extra Tips:



Easy Crop Recommendation for Farmers

Easy Crop Recommendation for Farmers

Nitrogen (N)	30	*(0–150)*
Phosphorus (P)	50	*(0–150)*
Potassium (K)	78	*(0–150)*
Temperature (°C)	20	*(10–40)*
Humidity (%)	34	*(20–90)*
Soil pH	5	*(4–9)*
Rainfall (mm)	300	*(50–300)*

Predict Crop

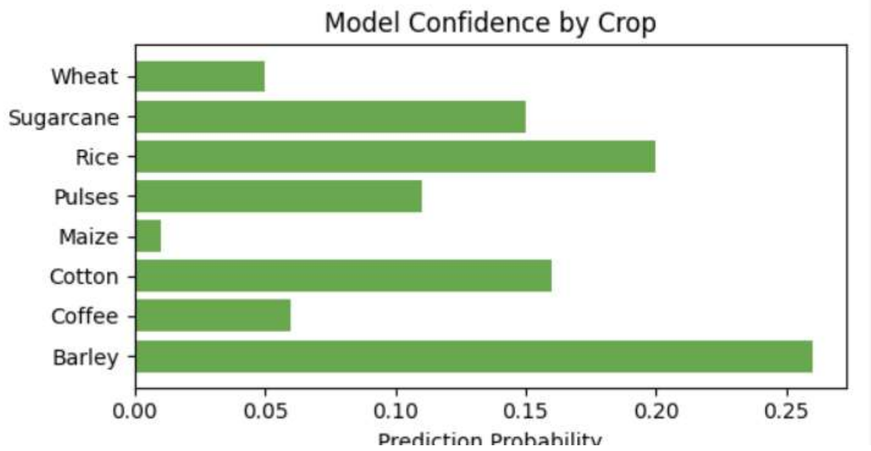
Or Search Crop Info: wheat

Search

Information for Wheat:

Wheat Cultivation Guide

- Cultivation: Sow in well-drained loamy soil; use certified seeds; maintain row spacing.
- Water: Moderate — critical during tillering & grain filling.
- Pests: Rusts, aphids; monitor during cool, humid weather.
- Harvest: Harvest when straw turns yellow and grains hard.
- Fertilizer: Basal N and top-dress at tillering; balanced NPK.
- Soil: Loamy soils with good drainage.
- Climate: Cool to moderate temperatures; less humid than rice.



Thank You

Questions?