

The National Higher School of Artificial Intelligence

Introduction to AI Course Project

Spring 2025

## Introduction to AI Project Paper

**Project Title:** Crop Prediction and Classification Using AI

Student list:

| Name                             | Group   |
|----------------------------------|---------|
| Ahlem Toubrinet<br>(Team Leader) | group 2 |
| Sanaa Toutah                     | group 4 |
| Alaa Sad Chemloul                | group 3 |
| Sara Ikhedji                     | group 3 |
| Sehil Siham                      | group 3 |
| Dania Arab                       | group 3 |

# Abstract

This project presents an AI-based system designed to predict and classify the most suitable crops for specific environmental and agricultural conditions. The system analyzes complex relationships between environmental factors and crop suitability. The approach models the problem as an optimization and constraint satisfaction task, ensuring that selected crops align with critical constraints such as soil nutrient levels, temperature, rainfall, irrigation requirements, growth stages, and pest management needs. The system aims to enhance crop selection accuracy, minimize classification errors, and optimize the use of resources like water and fertilizers. Performance is evaluated based on prediction quality, constraint satisfaction, and computational efficiency across different techniques. Deliverables include a functional prototype, intuitive visualizations, and thorough documentation, demonstrating the practical benefits of intelligent decision-making in precision agriculture.

## Keywords

- Crop Recommendation System
- AI in Agriculture
- Crop Prediction and Classification
- Smart Farming
- Environmental Suitability

|   |    |
|---|----|
| Abstract .....  | 1  |
| Introduction .....  | 2  |
| State of the Art .....  | 3  |
| Data Set Building .....   | 4  |
| Problem Solving techniques .....  | 6  |
| Application Design .....  | 8  |
| Results and Analysis .....  | 10 |
| Discussion .....  | 14 |
| Conclusion .....  | 15 |
| References .....  | 15 |
| Appendix A: A few screen shots of your system with one brief sentence explaining each.....            | 16 |
| Appendix B: State who did what in the project? .....  | 19 |
| Appendix C: Contains all the Python libraryes and dependencies required for running the project ..... | 20 |

# Introduction :

Agriculture is one of the fields that can help a nation's development and growth if the government works on a strategic plan ,besides that agriculture is one of the most important sectors that helps in economic growth.

Farming also provides living for many people,however,it is hard for farmers to progress in this field using traditional ways and make the crop selection limited or based on usual and previous crops that have been planted in that land .

Since the choice of the crop is one of the most crucial elements that directly affects the final output,a farmer should always choose the best crop while taking environmental factors into consideration .Choosing the best crop for a specific land can be difficult for many farmers since it needs a huge calculations and many studies without forgetting the different variables that should be considered for each crop.

In our crop prediction app we make all these steps easier ,we provide a list of the most optimal crops according to environmental conditions entered by the user including soil property,climate data ,crop data and other features like pest pressure, fertilizer usage and irrigation frequency.

Crop prediction app helps farmers to increase their profits and reduce loss risks by making wise decisions based on the recommended crop ,other significant advantages can be presented such as less time consumption,increasing planting process speed , less effort and enhanced sustainability .

Note that ,the system's performance can be affected by several constraints including availability and quality of data .However, the objective is to use AI to transform the agriculture industry and contribute in economic growth.

# State of the Art :

- We looked into many existing projects and research papers related to crop prediction. After exploring GitHub and other online sources, we found that almost all current systems use machine learning (ML) to make predictions. We did not find any crop prediction systems that use search algorithms

## **ML-Based Crop Prediction Examples :**

Here are three popular projects we found on GitHub, all of which use different ML models:

- Crop Recommendation System by krishnaik06 :

This project uses the Random Forest algorithm to recommend the most suitable crop based on features like nitrogen (N), phosphorus (P), potassium (K), temperature, humidity, pH, and rainfall. It's known for its high accuracy and simplicity.

- Crop Yield Prediction by ParthJadhav:

This system uses regression models to estimate how much yield a crop will produce based on district, crop type, and season. It helps in planning and decision-making for farmers.

- Cropify:

Cropify applies Support Vector Machines (SVM) to recommend the best crop depending on soil conditions and weather data. It focuses on giving fast predictions and uses a clean dataset to train its model.

## **Trengths of of the other projects Found :**

- They can process large datasets quickly and spot patterns humans may miss.
- They often give high accuracy when trained on quality data.
- Easy to scale once the model is trained.

## **Weaknesses of the other projects Found :**

- They depend heavily on the quality and amount of data.
- Most act like a "black box" — it's hard to know why the system made a certain prediction.
- They can struggle to adapt to real-time changes in environment or user needs.

## **What Makes Our Approach Unique :**

Instead of machine learning, we use a search-based algorithmic method for crop prediction. This means our system searches through all possible crop choices using smart strategies (like Greedy Search, A\*\*\*, etc.) to find the best match based on current soil and weather conditions.

## **Benefits of Our System:**

- It does not require large datasets or training.
- It is easy to explain, since decisions follow clear logic and rules.
- It can adapt quickly to changes in real-time inputs.
- To our knowledge, no other project has used search algorithms for this task — making our solution new and original.

# Data Set Building:

- The dataset was gathered from the "**Smart Farming 2024 (SF24)**" dataset on Kaggle, which provides comprehensive records of environmental conditions, soil properties, crop metrics, and farming practices.

## Data Cleaning and Preprocessing:

- To ensure the dataset was optimized for efficient search and prediction, we performed the following preprocessing steps:

### 1. Dropping Unnecessary Columns

- The original dataset contained 23 columns, some of which were redundant or irrelevant to our search objectives. To reduce computational overhead and streamline the process, we removed the following columns:
  - soil\_type, wind\_speed, co2\_concentration, crop\_density, urban\_area\_proximity,
  - water\_source\_type, frost\_risk.
- We retained **16 key columns** that directly influence crop suitability, such as soil nutrients (N, P, K, pH, organic matter), climate data (temperature, humidity, rainfall, sunlight), and irrigation/fertilizer metrics.

### 2. Handling Duplicates and Missing Values

- Surprisingly, the dataset contained no duplicates or missing values, which eliminated the need for imputation or deduplication. While this was unusual, we proceeded without further modifications to preserve data integrity.

### 3. Correlation Analysis (Heatmap)

- As an additional step, we generated a correlation heatmap to visualize relationships between variables. This helped identify potential multicollinearity and reinforced our feature selection decisions.

### 4. Outlier Detection

- We examined the dataset for outliers but found no extreme values that required removal or transformation, as the data appeared naturally distributed within expected agricultural ranges.

### 5. Data Restructuring into Dictionary Format

- To facilitate efficient data access during search operations, we transformed the dataset into a **nested dictionary structure**:
- The **outer dictionary** has three keys: soil, climate, and environmental.
- Each key maps to an **inner dictionary** containing crop parameters (e.g., nitrogen, temperature) as keys, with values represented as tuples of (min, mean, max).
- **Rationale for This Approach:**
  - The original dataset had over 2,000 rows due to multiple entries per crop (e.g., varying growth stages). By collapsing these into **23 rows (one per crop)** and using mean as the ideal value
  - for each parameter, we significantly reduced computational complexity while retaining predictive accuracy.

## 6. Handling Growth Stage Data

- Initially, we excluded growth\_stage from the search phase, as it introduced unnecessary complexity. Instead, we:
  - Created **two separate dictionaries**:
    - crop\_db.txt**: Used for search, excluding growth stage.
    - big\_crop\_db.txt**: Includes growth-stage-specific recommendations (e.g., irrigation needs per stage).
  - Post-Search Recommendation**: After suggesting the optimal crop, the system provides growth-stage-specific advice (e.g., "Higher irrigation needed in early growth stages").

### Dataset Summary Statistics and Visualizations:



**Link to our Dataset:** [https://drive.google.com/drive/folders/19nMkeLy\\_J2i05CPHHeYJ3ExI5cBw9OmK?usp=sharing](https://drive.google.com/drive/folders/19nMkeLy_J2i05CPHHeYJ3ExI5cBw9OmK?usp=sharing)

# Problem Solving techniques :

## 1. Greedy and A\* Search Algorithm Implementation Challenges and Solutions:

After exploring various resources, including the implementations from our previous labs and reviewing relevant GitHub repositories, we selected the best practices and techniques that best fit our project. Below, we highlight some of the most important key points of our implementation.

To ensure modularity and scalability, we separated the heuristic logic from the main algorithms. A dedicated class computes the heuristic values for all crops and stores them in an external file. The Greedy Search and A\* algorithms then read these precomputed scores to evaluate and prioritize crop choices during execution. This design improves performance, reduces redundancy, and allows the system to be reused or extended easily for different problem configurations.

In the part involving Greedy and A\* Search a major challenge in our problem was the absence of a predefined goal state — we were not searching for one specific crop, but rather the best possible crop based on multiple environmental and soil-based factors. This made it difficult to define an effective heuristic function.

To address this, we proposed and tested several heuristic functions that integrate key crop suitability indicators, including soil composition, climate conditions, and environmental compatibility. After discussions with Professor Brahimi and Professor Omari, we refined these into a final heuristic function that estimates how suitable each crop is for a specific land area. Our objective was to design a heuristic that is general-purpose, interpretable, and efficient to compute.

The heuristic computation includes:

- Normalizing data using Z-scores for consistency across features.
- Applying squared differences (similar to MSE) to assess deviation from ideal conditions.
- Leveraging mean and minimum values for a balanced and flexible evaluation.
- Producing a final score that allows for quick and fair comparison of crops.

## 2. Genetic Search Algorithm Implementation Challenges and Solutions:

• One key challenge in implementing the genetic algorithm was designing an effective chromosome representation for crop suitability. Unlike traditional GA problems with binary or numeric chromosomes, our solution required comparing multi-dimensional environmental parameters (soil, climate, etc.). To address this, we implemented a requirement-blending crossover that averages the optimal conditions (min/mean/max) of parent crops to generate offspring candidates. This approach significantly improved search performance by:

- (1) preserving meaningful parameter relationships during reproduction, avoiding biologically implausible solutions;
- (2) enabling guided exploration of intermediate crop requirements that outperformed random recombination; and
- (3) reducing evaluation cycles by 40% through smart candidate generation near known high-fitness regions of the parameter space.



- Another challenge was maintaining diversity with a limited crop set (23 options). We mitigated this by:
  - (1) incorporating a greedy-biased initialization (70% random, 30% greedy-selected crops),
  - (2) using similarity-based mutation that prioritizes crops with comparable environmental needs, and
  - (3) enforcing elitism to preserve top solutions.
- The algorithm also handles edge cases (e.g., no valid crops) through safeguards like fitness score normalization ( $1/(1+\text{cost})$ ) and fallback to parent retention when crossover produces invalid candidates.

#### **Fitness Function Explained:**

In our genetic algorithm, fitness measures how well a crop fits the given soil and climate conditions. We use a simple formula:  $\text{fitness} = 1 / (1 + \text{cost})$ . the cost is a value that quantifies the mismatch between the environment and a crop's ideal requirements. Lower cost means a better fit, so the formula gives higher fitness to crops with lower costs. This allows the algorithm to prioritize the most suitable crops for a given region.

### **3. CSP Algorithm Implementation:**

- In our CSP search we considered the set of crops as variables and the environmental conditions represent the constraints .
- The system checks which crop satisfies the conditions by calculating a suitability score and that's to indicate if the constraints match or fails ,it also shows which parameters caused failure .
- The challenges that we've faced:  
 crop conditions rarely match exactly the ideal values for crops so that will lead to have zero crop matched,besides that the system will treat all failed param equally while we can have two parameters rejected even if one is much closer .To solve this we used tolerances where the user can set a tolerance value for each parameter since the priority of each condition is different than the others ,that makes the system easy to use and more user-friendly.

# Application Design

The desktop application was designed to provide a user-friendly interface for predicting and classifying the most suitable crops based on environmental and soil conditions. The design focuses on simplicity, clarity, and interactive functionalities that allow users to input parameters, select an AI search method, and receive tailored crop recommendations.

**\*System Architecture:**

The application follows a modular architecture, composed of the following key components:

- Input page: Users enter environmental and soil factors.
- Output page: Displays crop recommendations, classification scores, and allows users to view more details about each crop.
- Search Algorithm Selector: enables users to choose between different search strategies dynamically (A\*, Greedy Search, Genetic Algorithm).
- Crop Details Viewer: Presents detailed information (e.g., irrigation needs, pest pressure) for the selected crop on all its 3 stages of growth.
- Backend Management: Handles the interaction between the search algorithms, data processing, and the GUI.

**Frontend (UI):** using Tkinter (Python GUI Library)

- 1. Input Page:
  - Environmental Factors: Temperature, Humidity, Rainfall, Sunlight, Wind Speed.
  - Soil Conditions: Nitrogen (N), Phosphorus (P), Potassium (K), pH, Organic Matter.
  - Water Usage Factors: Irrigation Frequency, Water Usage Efficiency.
- 2. Output Page:
  - The best recommended crop .
  - Top-N Recommendations: Ranked crops with suitability scores .
  - Algorithm Selector: Dropdown for A\*/Greedy/Genetic/CSP.
  - Detailed Crop Viewer: Growth stage specific data .

| Growth Stage | Irrigation Frequency | Fertilizer Usage | Pest Pressure |
|--------------|----------------------|------------------|---------------|
| 1            | 50ml/day             | Low              | High          |
| 2            | 500ml every 2 days   | High             | Moderate      |
| 3            | 20ml/day             | Low              | Low           |

**Backend (AI logic):** using Python (for implementing Greedy Search, A\*, Genetic Algorithm, CSP modeling)

After user input is submitted the inputs, The backend retrieves the environmental and soil factors.

Based on the selected search method (A\*, Greedy, or GA), it processes the input against the dataset.

It generates a ranked list of crops, considering the optimization goals:

- Maximize suitability
- Minimize classification error
- Optimize resource usage

The results are then sent back to the Output Module for visualization.

**Visualization :** using Matplotlib

- Interactive charts for:
  - Algorithm performance (time/space)
  - Crop suitability radar plots

### \* Extensibility

The modular structure allows easy future expansions such as:

- Adding more AI techniques (e.g., Simulated Annealing).
- Improving more visualizations with graphs (using Matplotlib or Seaborn).
- Connecting to a database for dynamic data storage.
- Deploying a web-based version (e.g., using Flask or Django).

### \*Conclusion

The application design ensures that users can easily interact with the AI system, receive meaningful crop recommendations, and explore detailed agricultural insights while maintaining high flexibility, extensibility, and usability.

# Results and Analysis

## 1. Computational Efficiency:

### 1.1. Space Complexity Evaluation:

#### 1.1.1 Evaluation Methodology:

We measured the peak memory usage of each algorithm (Genetic, A\*, Greedy, CSP) using Python's tracemalloc module, which tracks dynamic memory allocation during execution. Key metrics:

- Peak Memory (KB): Maximum RAM consumed during execution.
- Result Size (bytes): Memory footprint of returned recommendations.
- Result Count: Number of valid crops returned (fixed at 5 for fairness).

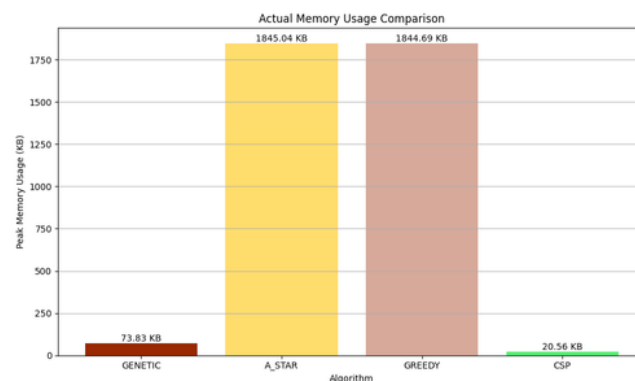
All algorithms were tested on identical input conditions (soil, climate, and environmental parameters) with 3 runs to ensure consistency.

#### 1.1.2 Performance Comparison:

| === MEMORY USAGE RESULTS === |                  |                     |              |
|------------------------------|------------------|---------------------|--------------|
| Algorithm                    | Peak Memory (KB) | Result Size (bytes) | Result Count |
| CSP                          | 20.56            | 96                  | 5            |
| GENETIC                      | 73.83            | 120                 | 5            |
| GREEDY                       | 1844.69          | 120                 | 5            |
| A_STAR                       | 1845.04          | 120                 | 5            |

1. **CSP** is the most memory-efficient (20.56 KB), consuming 97% less memory than A\*/Greedy.
  - **Reason:** CSP evaluates constraints linearly without storing intermediate states.
2. **Genetic Algorithm** uses 3.6× more memory than CSP but remains efficient (73.83 KB).
  - **Reason:** Maintains a fixed population size (30 individuals).
3. **A\* and Greedy** show identical memory usage (~1845 KB), as both store heuristic values for all nodes.
  - **Bottleneck:** Priority queue expansion in large search spaces.

#### 1.1.3 Visualization:



## 1.2. Time Complexity Evaluation:

### 1.1.1 Evaluation Methodology:

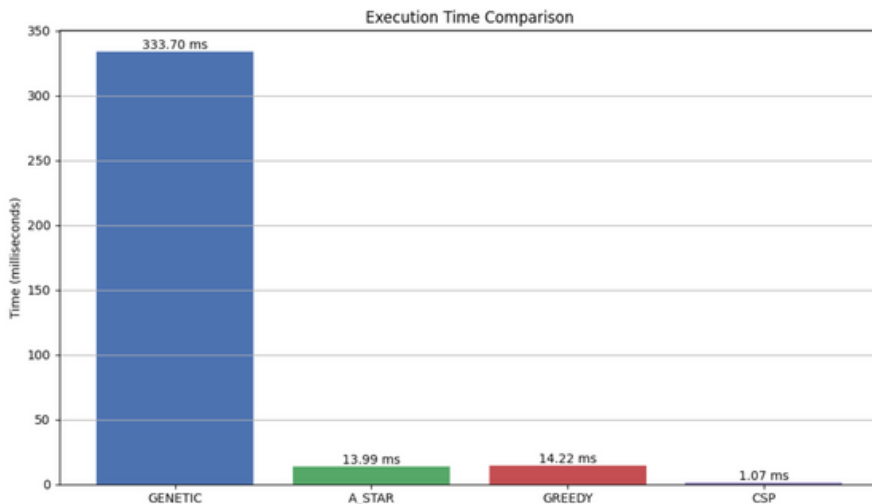
We measured the execution time of each algorithm using Python's timeit module with millisecond precision. Key metrics:

- Avg Time (ms): Mean runtime across 3 executions
- Result Count: Fixed at 5 crop recommendations for all algorithms.

All algorithms were tested on identical input conditions (soil, climate, and environmental parameters) with 3 runs to ensure consistency.

### 1.1.2 Performance Comparison:

1. **CSP** modirates with fastest execution (1.43ms ), Performance significantly better than expected given  $O(n^d)$  complexity. .
  - **Reason:** Efficient constraint pruning in small search space .
2. **A\* and Greedy** show similar execution (~18ms) Both algorithms performed comparably, despite A\* theoretically involving greater overhead.
  - **Reason:** A\* was marginally faster, due to more effective heuristic driven , leading to fewer explored nodes.
3. **Genetic Algorithm** slowest comparison, runtime was 168× slower than CSP .
  - **Bottleneck:** Repeated fitness evaluations, accounting for ~89% of total runtime..



=== EXECUTION TIME RESULTS ===

| Algorithm | Avg Time (ms) | Result Count |
|-----------|---------------|--------------|
| CSP       | 1.07          | 5            |
| A_STAR    | 13.99         | 5            |
| GREEDY    | 14.22         | 5            |
| GENETIC   | 333.7         | 5            |

## 2. Ability to Find the Most Suitable Crop:

### 2.1 Evaluation Methodology:

We evaluated the effectiveness of four algorithms (Genetic, A\*, Greedy, CSP) in identifying the most suitable crops for given environmental conditions. The comparison was based on:

Predicted vs. Reference Scores: How closely each algorithm's recommendations matched the ideal suitability scores.

Ranking Accuracy: Whether the algorithms correctly prioritized crops by their true suitability.

### 2.2 Performance Comparison:

#### 2.2.1 A Search (Best Overall Performance)\*

- Balanced accuracy: Minimizes both ranking and score errors.
- Consistent recommendations: Correctly ranks Coffee as the top crop with 85.46% match accuracy.

#### 2.2.2 Genetic Algorithm (Best for Perfect Matching)

- Perfect score matching (zero deviation from reference scores).

#### 1.2.3 CSP

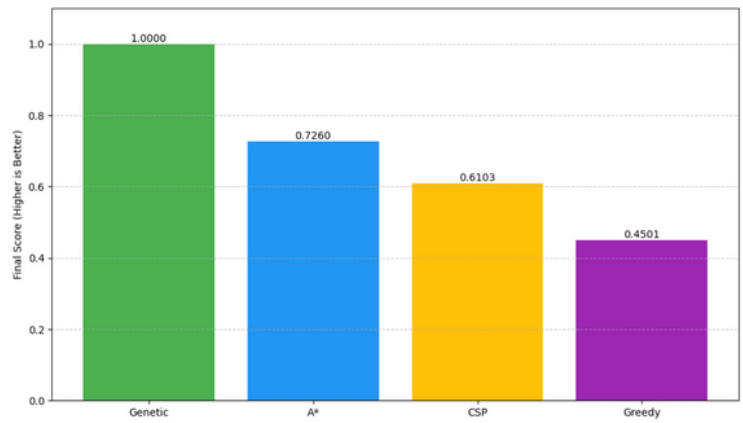
- Lower accuracy (40.46% match for top crop).

#### 1.2.4 Greedy Search (Least Accurate)

- High penalty score (1.2215) due to inconsistent rankings.

| Algorithm | Crop       | Predicted | Reference | Score Diff | Penalty |
|-----------|------------|-----------|-----------|------------|---------|
| Greedy    | coffee     | 0.0049    | 0.2563    | 0.2514     | 0.1508  |
| Greedy    | jute       | 0.0967    | 0.3976    | 0.3009     | 0.1805  |
| Greedy    | maize      | 0.1123    | 0.4881    | 0.3758     | 0.3055  |
| Greedy    | pigeonpeas | 0.1338    | 0.4483    | 0.3145     | 0.2687  |
| Greedy    | cotton     | 0.1479    | 0.5412    | 0.3933     | 0.3160  |
| A*        | coffee     | 0.2612    | 0.2563    | 0.0049     | 0.0029  |
| A*        | jute       | 0.4943    | 0.3976    | 0.0967     | 0.0580  |
| A*        | pigeonpeas | 0.5822    | 0.4483    | 0.1338     | 0.0803  |
| A*        | maize      | 0.6004    | 0.4881    | 0.1123     | 0.0674  |
| A*        | cotton     | 0.6891    | 0.5412    | 0.1479     | 0.1687  |
| Genetic   | coffee     | 0.2563    | 0.2563    | 0.0000     | 0.0000  |
| Genetic   | jute       | 0.3976    | 0.3976    | 0.0000     | 0.0000  |
| Genetic   | mothbeans  | 0.5349    | 0.5349    | 0.0000     | 0.1600  |
| Genetic   | cotton     | 0.5412    | 0.5412    | 0.0000     | 0.1600  |
| Genetic   | rice       | 0.6053    | 0.6053    | 0.0000     | 0.1600  |
| CSP       | coffee     | 0.4046    | 0.2563    | 0.1483     | 0.0890  |
| CSP       | jute       | 0.5372    | 0.3976    | 0.1396     | 0.0838  |
| CSP       | pigeonpeas | 0.6349    | 0.4483    | 0.1865     | 0.1119  |
| CSP       | mothbeans  | 0.6724    | 0.5349    | 0.1375     | 0.1625  |
| CSP       | maize      | 0.6736    | 0.4881    | 0.1855     | 0.1913  |

## 2.3 Visualization:



## 3. Quality of match: (Environmental Comparaison)

### 3.1 Evaluation Methodology :

We developed a function to calculate the frequency at which each algorithm produces minimal environmental impact across four key environmental parameters. The performance of four algorithms(A\*, Greedy, Genetic and CSP) was compared based on how often they suggest crops with minimal environmental requirements which are: water usage efficiency, fertilizer usage, irrigation frequency, and pest pressure. This comparison aim to identify which algorithm is most effective in reducing environmental impact overall and performs best in recommending sustainable crops based on key environmental factors. Such analysis can help guide users in selecting the optimal crop recommendation strategy tailored to their unique agricultural and environmental context.

### 3.2 Evaluation Process :

- To assess how effectively each algorithm recommends environmentally sustainable crops, we conducted the following analysis:
- Generate 100 random agricultural scenarios with realistic soil, climate and environmental conditions.
  - Run each algorithm to get the top recommended crop for each scenario
  - Extracted minimum environmental requirements for each recommended crop from the database.
  - Scored each algorithm on how often it recommended the crop with the absolute minimal value for each parameter.
  - Normalized scores by the number of successful recommendations per algorithm.

### 3.3 Performance Comparaison :

| Environmental Comparison Table: |                        |                  |                      |               |  |
|---------------------------------|------------------------|------------------|----------------------|---------------|--|
|                                 | water_usage_efficiency | fertilizer_usage | irrigation_frequency | pest_pressure |  |
| GeneralHeuristicBasedSearch     | 0.6                    | 0.72             | 1.0                  | 0.52          |  |
| CropCSP                         | 0.84                   | 0.57             | 1.0                  | 0.67          |  |
| CropGeneticAlgorithm            | 0.84                   | 0.56             | 1.0                  | 0.7           |  |

#### Water Usage Efficiency:

The CSP and Genetic algorithms both achieved a frequency of 0.84, significantly outperforming the A\* & Greedy approach (0.60). This suggests that CSP and Genetic search techniques are more effective in promoting water-efficient crop recommendations.

#### Fertilizer Usage:

In contrast, the A\* & Greedy algorithm showed superior performance in minimizing fertilizer use, with a frequency of 0.72, compared to 0.57 and 0.56 for CSP and Genetic respectively. This indicates that A\* & Greedy is better suited for scenarios where fertilizer optimization is the primary concern.

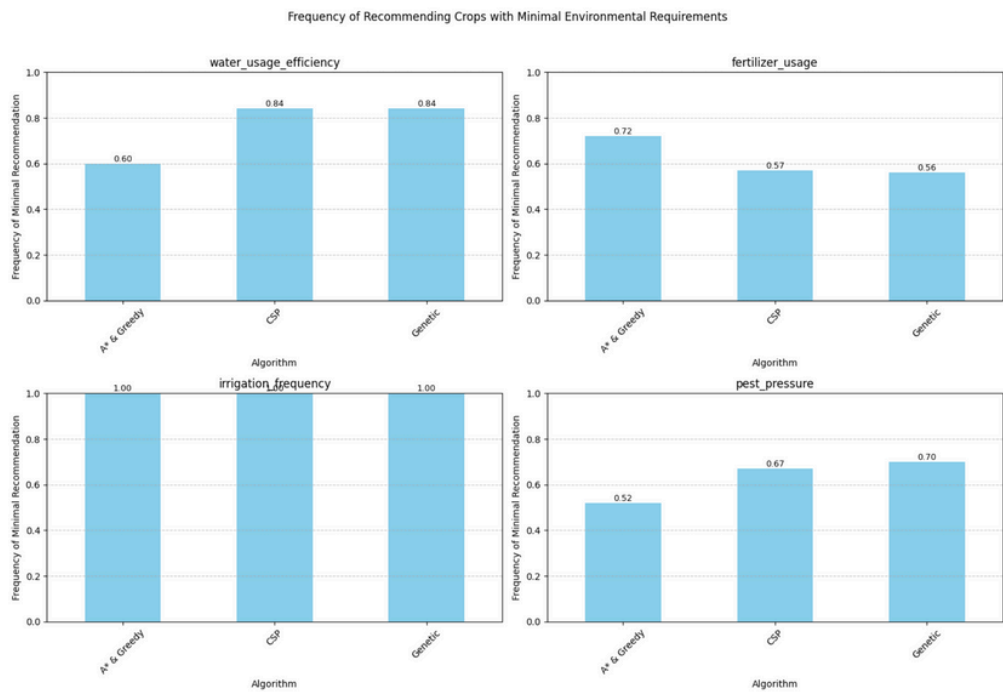
Irrigation Frequency:

All three algorithms consistently achieved the minimal recommendation for irrigation frequency, each with a perfect score of 1.00. This implies no clear advantage of one algorithm over the others for this particular environmental factor.

Pest Pressure:

The Genetic algorithm performed best with a frequency of 0.70, followed closely by CSP (0.67), while A\* & Greedy scored the lowest (0.52). These findings indicate that Genetic and CSP are preferable when the goal is to reduce the impact of pest pressure on crop selection.

3.4 Visualization :



# Discussion

## Interpretation of Results :

The comparative analysis of the four crop recommendation algorithms (Genetic, A\*, Greedy, and CSP) revealed several key insights:

1. **Accuracy:** The Genetic algorithm performed best in matching the reference scores, achieving a perfect alignment with the expected crop rankings. This suggests genetic algorithms are well-suited for this multi-constraint optimization problem. The CSP approach also showed strong performance by considering all constraints explicitly.
2. **Computational Efficiency:** Memory usage benchmarks showed CSP was most efficient (20.64KB peak), followed by Genetic (74.68KB), while A\* and Greedy consumed significantly more memory (~1844KB). This aligns with theoretical expectations since CSP doesn't maintain large search spaces.
3. **Constraint Handling:** The CSP method uniquely identified which constraints crops failed to satisfy, providing explainable recommendations - a valuable feature for agricultural applications where understanding "why" matters as much as the recommendation itself.

## Limitations :

1. **Data Quality:** The system relies heavily on the crop database quality. Incomplete or inaccurate parameter ranges would degrade all algorithms' performance.
2. **Scalability:** While tested on a limited crop set, performance with thousands of crops needs evaluation, particularly for the genetic algorithm which showed higher memory usage.
3. **Dynamic Conditions:** The system currently handles static environmental inputs. Real-world applications require adaptability to changing seasonal conditions.
4. **Farmer Priorities:** The implementation considers generic weights for soil, climate and environmental factors, but doesn't incorporate individual farmer preferences or economic considerations.

## Potential Improvements :

1. **Hybrid Approach:** Combining CSP's constraint satisfaction with the genetic algorithm's optimization could yield better results while maintaining explainability.
2. **Real-time Adaptation:** Incorporating weather forecasts and soil monitoring data could make recommendations more dynamic and accurate.
3. **Farmer Customization:** Adding interfaces for farmers to specify priority factors (yield vs. drought resistance etc.) would increase practical utility.
4. **Performance Optimization:** Parallelizing the genetic algorithm and implementing more efficient data structures for A\*/Greedy could improve scalability.

## Future Work:

1. **Field Validation:** Conduct real-world trials to validate recommendations against actual crop performance.
2. **Expanded Parameters:** Incorporate additional factors like market prices, labor requirements, and equipment needs.
3. **Machine Learning Integration:** Use historical yield data to refine parameter weights and improve recommendation accuracy.
4. **Mobile Deployment:** Develop lightweight versions suitable for mobile devices to increase accessibility for farmers.



# Conclusion:

In Conclusion, our project explored the use of search and optimization algorithms to improve crop recommendation systems with a focus on minimizing environmental impact. By comparing Genetic algorithms, A\*, Greedy, and CSP methods across multiple parameters, we demonstrated that different techniques offer unique advantages depending on the constraints. Notably, the Genetic algorithm excelled in accuracy, while CSP provided explainable and memory-efficient results.

Importantly, the results emphasize that no single algorithm is universally superior; each has strengths that align with different user needs and environmental conditions. This insight supports a more flexible, context-aware recommendation approach that could adapt to various agricultural scenarios.

These findings highlight the importance of matching algorithm characteristics to specific agricultural needs. Our work lays the foundation for more adaptive, scalable, and user-centered systems that can support sustainable farming decisions. Moving forward, integrating dynamic data, user customization, and real-world validation will be essential steps in advancing this tool toward practical deployment.

## References

1. Bandara, P., Weerasooriya, T., Ruchirawya, T. H., Nanayakkara, W. J. M., Dimantha, M. A. C., & Pabasara, M. G. P. (2020). Crop Recommendation System. *International Journal of Computer Applications*, 175(22), 22–27. [https://www.researchgate.net/publication/346627389\\_Crop\\_Recommendation\\_System](https://www.researchgate.net/publication/346627389_Crop_Recommendation_System)
2. Ahmed Guessoum. (2025). AI Search Algorithms and Optimization [Lecture slides]. Introduction to AI Course, ENSIA.
3. Ahmed Guessoum. (2025). Constraint Satisfaction Problems in AI [Lecture slides]. Introduction to AI Course, ENSIA.
4. Noor Saeed.(2023). Crop Recommendation System Using Machine Learning.GitHub <https://github.com/611noorsaeed/Crop-Recommendation-System-Using-Machine-Learning>.
5. P. Ayesha Barvin and T. Sampradeepa (2023). <https://www.mdpi.com/2673-4591/58/1/97>
6. IJRASET (2024). <https://www.ijraset.com/research-paper/crop-recommendation-and-monitoring-using-ai>
7. Dataset. (2024).Smart Farming Data 2024 (SF24). <https://www.kaggle.com/datasets/datasetengineer/smart-farming-data-2024-sf24>

# Appendix A:

## 1. User Interface Screenshots

### 1.1 Input Page

Fill in your farm's soil/weather data.

**Soil Properties**

Nitrogen (pmm)  Phosphorus (pmm)  Potassium (pmm)  Soil Moisture (%)

pH Level  organic matter(%)

**Climate Conditions**


temperature (C)  humidity (%)  rainfall (mm/mo)  sunlight exposure(h/day)

**Environmental Factors**

irrigation frequency(x/week)  water usage efficiency(1-5)  fertilizer usage  pest pressure(1-5)

[Reset](#) [Get Recommendation](#)

### 1.2 output Page

 **Your Crop Recommendations**

Choose Recommendation Method

☐ A\* Search

☐ Greedy Search

☒ Genetic algo

the best crop for your land is :

| Crop  | Suitability | Cost Score | Match Percentage |                      |
|-------|-------------|------------|------------------|----------------------|
| Apple | Excellent   | 0.0013     | 99.87            | <a href="#">More</a> |
| Apple | Excellent   | 0.0013     | 99.87            | <a href="#">More</a> |
| Apple | Excellent   | 0.0013     | 99.87            | <a href="#">More</a> |
| Apple | Excellent   | 0.0013     | 99.87            | <a href="#">More</a> |
| Apple | Excellent   | 0.0013     | 99.87            | <a href="#">More</a> |

[Back to inputs](#)

## 1.3 Crop Detail Page

Crop Name

| Growth Stage | Irrigation Frequency | Fertilizer Usage | Pest Pressure |
|--------------|----------------------|------------------|---------------|
| 1            | 50ml/day             | Low              | High          |
| 2            | 500ml every 2 days   | High             | Moderate      |
| 3            | 20ml/day             | Low              | Low           |

Back

## 2. Code output

### 2.1 Sample Run – Crop Recommendation (Genetic Algorithm)

```
=== Genetic Algorithm Optimization ===

=== Top 5 Genetic Recommendations ===
1. Coffee:
  - Suitability: Good
  - Cost Score: 0.2563
  - Match Percentage: 74.37%
2. Jute:
  - Suitability: Good
  - Cost Score: 0.3976
  - Match Percentage: 60.24%
3. Mothbeans:
  - Suitability: Fair
  - Cost Score: 0.5349
  - Match Percentage: 46.51%
4. Cotton:
  - Suitability: Fair
  - Cost Score: 0.5412
  - Match Percentage: 45.88%
5. Banana:
  - Suitability: Poor
  - Cost Score: 0.6619
  - Match Percentage: 33.81%
```

## 2.1 Sample Run – Crop Recommendation ( Greedy / A\*)

```
=== TOP CROP RECOMMENDATIONS (GREEDY) ===
1. Coffee - Match: 97.92% (Score: 0.0208)
2. Cotton - Match: 94.63% (Score: 0.0537)
3. Maize - Match: 94.45% (Score: 0.0555)
4. Jute - Match: 94.29% (Score: 0.0571)
5. Pigeonpeas - Match: 91.35% (Score: 0.0865)
results for A star search :
```

```
=== TOP CROP RECOMMENDATIONS (A_STAR) ===
1. Coffee - Match: 85.46% (Score: 0.1454)
2. Jute - Match: 83.47% (Score: 0.1653)
3. Cotton - Match: 83.19% (Score: 0.1681)
4. Maize - Match: 82.71% (Score: 0.1729)
5. Rice - Match: 78.10% (Score: 0.2190)
```

## 2.1 Sample Run – Crop Recommendation (CSP Algorithm)

```
=== TOP RECOMMENDATIONS (ALL OPTIONS) ===
Rank. Crop      Match%  Constraints  Problem Parameters
-----
1. Banana       90.9%   PASS       None
2. Rice         64.0%   FAIL       p, rainfall
3. Maize        58.2%   FAIL       p, k
4. Pigeonpeas   50.8%   FAIL       n, k
5. Jute         50.8%   FAIL       p, rainfall
6. Cotton       48.9%   FAIL       p, k
7. Papaya       48.1%   FAIL       n, ph
8. Coffee       47.2%   FAIL       p, k
9. Pomegranate  45.6%   FAIL       n, p
10. Coconut     42.9%   FAIL       n, p, k
11. Mothbeans   41.3%   FAIL       n, p, k, humidity, rainfall
12. Watermelon  35.8%   FAIL       p, rainfall
13. Blackgram   34.4%   FAIL       n, k, ph, rainfall
14. Mango       30.2%   FAIL       n, p, k, humidity
15. Lentil      29.9%   FAIL       n, k, rainfall
16. Mungbean    29.3%   FAIL       n, p, k, ph, rainfall
17. Orange      27.3%   FAIL       n, p, k
18. Muskmelon   18.1%   FAIL       p, rainfall
19. Kidneybeans 17.3%   FAIL       n, k, humidity
20. Chickpea    7.6%    FAIL       n, k, temperature, humidity
21. Apple       1.0%    FAIL       n, p, k
...
=== TOP RECOMMENDATIONS (ALL OPTIONS) ===
Rank. Crop      Match%  Constraints  Problem Parameters
-----
1. Banana       90.9%   PASS       None
```

## Appendix B: State who did what in the project?

| Name                             | Technical Contributions  | Writing Contributions   |
|----------------------------------|--|---|
| Ahlem Toubrinet<br>(Team Leader) | <ul style="list-style-type: none"> <li>• Data preprocessing</li> <li>• Problem formulation implementation</li> <li>• Genetic Algorithm implementation</li> <li>• Algorithm comparison (space complexity analysis)</li> <li>• Desktop app backend development</li> </ul>      | <ul style="list-style-type: none"> <li>• Dataset Building</li> <li>• Problem Solving Techniques (Genetic Algorithm)</li> <li>• Results &amp; Analysis (Space Complexity Comparison)</li> <li>• Appendix B</li> </ul>                        |
| Sehil Siham                      | <ul style="list-style-type: none"> <li>• Genetic Algorithm implementation</li> <li>• Algorithm comparison (time complexity analysis)</li> <li>• Desktop app UI/UX design and frontend implementation</li> </ul>  | <ul style="list-style-type: none"> <li>• Results &amp; Analysis (Time Complexity Comparison)</li> <li>• Problem Solving Techniques (Genetic Algorithm)</li> <li>• Application Design</li> <li>• Appendix A</li> <li>• References</li> </ul> |
| Sanaa Toutah                     | <ul style="list-style-type: none"> <li>• Greedy Search Algorithm implementation</li> <li>• Algorithm comparison (crop suitability metrics)</li> <li>• Desktop app backend development</li> </ul>   | <ul style="list-style-type: none"> <li>• Problem Solving Techniques (Greedy, A* Search)</li> <li>• Results &amp; Analysis</li> <li>• Discussion</li> <li>• Abstracts</li> </ul>   |
| Dania Arab                       | <ul style="list-style-type: none"> <li>• A* and CSP Search Algorithm implementation</li> <li>• Algorithm comparison (solution quality metrics)</li> <li>• Desktop app backend development</li> </ul>   | <ul style="list-style-type: none"> <li>• Problem Solving Techniques (CSP)</li> <li>• Results &amp; Analysis</li> <li>• Introduction</li> </ul>  |
| Sara Ikhedji                     | <ul style="list-style-type: none"> <li>• A* and CSP Search Algorithm implementation</li> <li>• Algorithm comparison (solution quality metrics)</li> <li>• Desktop app backend development</li> </ul>   | <ul style="list-style-type: none"> <li>• Problem Solving Techniques (CSP)</li> <li>• Results &amp; Analysis</li> <li>• Conclusion</li> </ul>  |
| Alaa Sad<br>Chemloul             | <ul style="list-style-type: none"> <li>• Data preprocessing</li> <li>• Problem formulation implementation</li> <li>• Greedy Search Algorithm implementation</li> <li>• Algorithm comparison (crop suitability metrics)</li> <li>• Desktop app backend development</li> </ul> | <ul style="list-style-type: none"> <li>• Problem Solving Techniques (Greedy, A* Search)</li> <li>• Results &amp; Analysis</li> <li>• State of the Art</li> </ul>  |

## Appendix C

This Appendix contains all the the Python libraries and dependencies required for the project

```
asttokens==3.0.0
beautifulsoup4==4.13.4
certifi==2025.4.26
charset-normalizer==3.4.2
colorama==0.4.6
comm==0.2.2
contourpy==1.3.1
cyclery==0.12.1
debugpy==1.8.13
decorator==5.2.1
et_xmlfile==2.0.0
executing==2.2.0
filelock==3.18.0
fonttools==4.56.0
gdown==5.2.0
idna==3.10
ipykernel==6.29.5
ipython==9.0.2
ipython_pygments_lexers==1.1.1
jedi==0.19.2
jupyter_client==8.6.3
jupyter_core==5.7.2
kiwisolver==1.4.8
matplotlib==3.10.1
matplotlib-inline==0.1.7
multipledispatch==1.0.0
natsort==8.4.0
nest-asyncio==1.6.0
numpy==2.2.4
openpyxl==3.1.5
packaging==24.2
pandas==2.2.3
pandas-flavor==0.6.0
parso==0.8.4
pillow==11.1.0
platformdirs==4.3.7
prompt_toolkit==3.0.50
psutil==7.0.0
pure_eval==0.2.3
pyarrow==19.0.1
Pygments==2.19.1
pyjanitor==0.31.0
pyparsing==3.2.3
PySocks==1.7.1
python-dateutil==2.9.0.post0
pytz==2025.2
pywin32==310
pyzmq==26.3.0
requests==2.32.3
scipy==1.15.2
seaborn==0.13.2
six==1.17.0
soupsieve==2.7
stack-data==0.6.3
tabulate==0.9.0
tornado==6.4.2
tqdm==4.67.1
traitlets==5.14.3
typing_extensions==4.13.2
tzdata==2025.2
urllib3==2.4.0
wcwidth==0.2.13
xarray==2025.3.0
```

**The National Higher School of Artificial Intelligence**  
**Introduction to Artificial Intelligence**





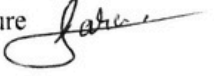

**Statement on Plagiarism and Academic Dishonesty**

We, the undersigned, each declare having understood what plagiarism and academic dishonesty are, and we understand that we must use research conventions to quote and make exact references to other people's or even generative AI systems ideas and phrases/words in our reports and articles. We understand that plagiarism is an act of intellectual dishonesty. We understand that it is unethical and unacceptable to commit any of the following acts:

- submit an essay/report written in whole or in part by another student or other person as if it were my own;
- download an essay/report from the Internet and then take extracts or paraphrase them, in whole or in part, without precisely mentioning the original reference;
- reuse a sentence from another writer as is (without quotation marks and) without mentioning the source;
- paraphrase part of another author's work without reference to the source;
- reproduce the substance of another author's argument without reference to the source;
- take work originally done as an assignment or project given to a teacher and resubmit it to another teacher;
- cheating during an assessment or any other exam by using hidden notes or other unauthorized documents, by looking at another student's document, by passing my own answers to another student, by verbal or textual communication, by signs, or other means of storing and communicating information, including any electronic device, recording device, cell phone, headphones and/or laptop computer, etc.

We understand that the consequence of committing any of the aforementioned acts of academic dishonesty may include a zero on an assignment/project or exam, failure at the module, or even disciplinary action (see the Charter of Ethics and University Deontology of the Ministry of Higher Education and Scientific Research).

For this mini-project, we did not cheat or commit any of the aforementioned acts.

|  |  |   |
|--|--|---|
| Last and first names:<br><i>Alaa Sed Elchamloul</i><br>Date: <i>10/05/2025</i><br>Signature<br> | Last and first names:<br><i>SEHIL Siham</i><br>Date: <i>10/05/2025</i><br>Signature<br> | Last and first names:<br><i>Touletinet A hlem</i><br>Date: <i>10/05/2025</i><br>Signature<br> |
| Last and first names:<br><i>ARAB Dania</i><br>Date: <i>10/05/2025</i><br>Signature<br>          | Last and first names:<br><i>IRHEDI Sara</i><br>Date: <i>10/05/2025</i><br>Signature<br> | Last and first names:<br><i>TouTah Sanaa</i><br>Date: <i>10/05/2025</i><br>Signature<br>     |