**🔍 Recon Prioritization AI (Server-Side) — Step by Step Plan**

**Step 1: Define the Scope Clearly**

* Goal: Given recon data from the payload (e.g., files, processes, open ports), rank them by **sensitivity**.
* MVP Requirement: AI should take a file/process sample and output one of:
  + High (likely sensitive)
  + Low (maybe interesting)
  + None (ignore)

**Step 2: Decide Features to Collect (Inputs)**

Focus on simple, extractable metadata:

* **File name keywords** → (password, secret, config, ssh).
* **File extension** → (.txt, .docx, .log, .dll).
* **File size** → Very small (likely junk), very big (media files) → low priority.
* **Process names** → (chrome.exe, svchost.exe, notepad.exe).
* **Port numbers** → (22/80/443 = interesting; 49152+ = noise).

👉 For MVP, start with just **file metadata** (name, extension, size).

**Step 3: Build Synthetic Dataset**

Make a small dataset manually (~200 rows). Example:

| **File Name** | **Extension** | **Size (KB)** | **Label** |
| --- | --- | --- | --- |
| passwords.txt | txt | 2 | High |
| id\_rsa | none | 4 | High |
| notes.docx | docx | 120 | Low |
| temp123.log | log | 500 | None |
| config.json | json | 3 | High |
| movie.mp4 | mp4 | 50000 | None |

* Collect real file samples from your system.
* Label them manually as High/Low/None.

**Step 4: Choose Model & Features**

* Convert metadata into features:
  + extension → one-hot encoded (txt=1, docx=0, etc.).
  + name\_keywords → binary flag (1 if keyword like “password”, “config”).
  + size → numeric bucket (small/medium/large).
* Algorithm:
  + **Random Forest Classifier** (handles mixed features easily).
  + Logistic Regression also works, but Random Forest gives better interpretability for this.

👉 Random Forest = best balance of accuracy & simplicity.

**Step 5: Train the Model**

1. Prepare dataset (CSV).
2. Encode categorical features (extensions, keywords).
3. Train Random Forest with scikit-learn.
4. Save model as recon\_priority\_model.pkl.

**Step 6: Test the Model**

* Run it on unseen file names:
  + Input: "db\_config.yaml" → Output: High.
  + Input: "video.mp4" → Output: None.
  + Input: "meeting\_notes.docx" → Output: Low.

Check accuracy with a test split.

**Step 7: Integrate with C2 Server**

On the C2 server, after payload uploads recon results (file list):

1. Parse recon results into features.
2. Load recon\_priority\_model.pkl.
3. Predict sensitivity for each file.
4. Return ranked list to operator or feed it into **Attack Decision AI**.

**Step 8: Final MVP Demo**

* Run payload on your PC → it uploads a list of discovered files.
* C2 runs Recon AI → assigns scores.
* Example Output:
  + passwords.txt → High
  + notes.docx → Low
  + temp123.log → None

**✅ End Result**

* **Input:** Recon data (file/process metadata).
* **Output:** Sensitivity class (High, Low, None).
* **Usage:** Helps decide what to attack/exfiltrate.
* **Deliverables for college:**
  + Dataset (CSV of files + labels).
  + Training notebook (Random Forest).
  + Trained model file.
  + C2 integration code that ranks recon findings.