**Block 1: Data Preprocessing & Splitting**

1. **Data Acquisition & Inspection**
   * **Input:**
     + **Obfuscated-MalMem2022.csv with 55 numeric features (e.g., pslist.nproc, handles.nkey, etc.), a binary label (is\_malware), and for malware rows a malware\_type.**
   * **Steps:**
     + **Load using Pandas.**
     + **Inspect with df.head() and df.info().**
     + **Check for missing values with df.isna().sum().**
   * **Decisions:**
     + **If <5% missing, drop rows; if more, impute using column mean.**
2. **Outlier Detection & Handling**
   * **Method:**
     + **Cap each numeric column at the 99th percentile (e.g., for pslist.nproc, if 99th percentile = 55, any value above 55 is capped).**
   * **Code:**

**Detailed Explanation Recap**

**Block 1: Data Preprocessing & Splitting**

* **Acquisition & Cleaning:**  
  The system loads the CSV (Obfuscated-MalMem2022.csv) containing 55 numeric features along with binary and multi-class labels. Missing values are either dropped or imputed, and extreme values are capped at the 99th percentile.
* **Feature & Label Separation:**  
  Features (55 columns) are isolated from labels. The binary label (is\_malware) distinguishes benign from malware, and the malware type is used for multi-class classification on malware rows.
* **Scaling:**  
  All features are scaled using StandardScaler (mean=0, std=1) and saved as scaler.pkl to ensure consistency during training and inference.
* **Train-Test Split:**  
  The data is split 70/30, with benign samples isolated for autoencoder training. The admin can upload new data to update this process, which re-runs the cleaning, scaling, and splitting, and saves an updated scaler.

**Block 2: Model Training**

* **Autoencoder:**  
  A neural network with a structure of 55→32→16→32→55 is trained on benign data (100 epochs, batch size 32, Adam optimizer with 0.001 learning rate). Its loss is measured by MSE, and a threshold (default ~0.05) is computed as mean error plus three times the standard deviation. The admin can adjust the threshold, epochs, and learning rate as needed.
* **Random Forest Classifiers:**  
  RF #1 (binary) is trained on the full training set with 100 trees, no maximum depth, using the "gini" criterion to distinguish benign from malware. RF #2 (multi-class) is trained on malware-only data to classify the specific malware type. Both models are saved as rf\_binary.pkl and rf\_maltype.pkl, respectively, with hyperparameters adjustable by the admin.

**Block 3: Storage & Logging**

* **Artifacts:**  
  All models, the scaler, and threshold values are stored on disk (or in a database) along with version information.
* **Logging:**  
  Every scan's details—including input features, autoencoder error, RF predictions, and computed risk rating—are logged in a database table (predictions\_log), facilitating future analysis and risk rating computation.

**Block 4: Risk Rating & Report Generation**

* **Risk Rating:**  
  The system computes a risk score for each scan using a weighted formula:

Risk=α×current error+(1−α)×historical error\text{Risk} = \alpha \times \text{current error} + (1-\alpha) \times \text{historical error}Risk=α×current error+(1−α)×historical error

with a default α=0.5\alpha = 0.5α=0.5. This rating is scaled (e.g., 0–10 scale) and displayed to the user. The admin can modify α\alphaα if needed.

* **Report Generation:**  
  Personalized reports (in PDF or CSV) are generated per scan, including all relevant details (features, error, predictions, risk rating, recommendations) and made available for download via a dedicated endpoint.

**Block 5: Web Application & Admin Interface**

* **User Flow:**  
  Users log in, upload files/hashes, and the system extracts features, scales them, and runs them through the hybrid prediction pipeline. The final JSON response includes the prediction result, risk rating, and a link to download the personalized report.
* **Admin Flow:**  
  Admins log in with elevated rights and have an interface to upload new datasets and adjust critical parameters (reconstruction error threshold, model hyperparameters, risk rating weight). When changes are made, they can trigger a full re-training of the models, which re-runs Blocks 1–3 and updates the artifacts and logs. An analytics dashboard provides real-time insights into predictions and system performance.