Inventory-Level Triage for Android Logical Images: Automated Artifact Type Classification

Rahib Aghababayev
Azerbaijan Technical University, Baku, Azerbaijan
rahib.agababayev@aztu.edu.az

***Abstract*— This study examines how accurately artifacts can be automatically identified and classified by type within an Android OS version 15 logical image. We propose a lightweight triage pipeline that derives features from file metadata, MIME/type hints, and path context, then applies supervised learning to assign coarse artifact types as Image, Audio, Document, Log/Config, App Cache/Object. To avoid duplicate leakage, evaluation uses a SHA-256 grouped split on 348 labeled files. On the leakage-aware test set, Logistic Regression and Random Forest achieve macro-F1=0.5808 and weighted-F1=0.9706, outperforming a rule-based baseline macro-F1=0.2496.**

Keywords*— *mobile forensics; android forensics; cyber sovereignty; artifact triage; logical acquisition; file inventory; metadata-based classification.

I. INTRODUCTİON

Mobile devices have become primary carriers of evidentiary traces in cyber incidents, fraud investigations, and cases involving critical services. In parallel, rapid digital transformation has elevated the importance of cyber sovereignty, where sensitive information, particularly in governmental or critical-infrastructure contexts - should be processed within trusted national environments and under defensible procedures. In practice, investigators often begin with a logical acquisition that yields a heterogeneous file inventory spanning media, documents, logs, app caches, and system-generated objects. The scale and heterogeneity of these inventories can slow early-stage review and delay time-to-insight during incident response and forensic workflows.

Android OS version 15 further intensifies this challenge. Evolving storage behaviors, app sandboxing, and fast-changing app layouts reduce the reliability of purely rule-based typing and increase dependence on tool- and app-specific parsers. While artifact parsing frameworks remain essential, logical exports frequently contain substantial content that is not immediately parsable (or only partially exported), leaving a practical gap: investigators still need an inventory-level triage layer that can categorize files without deep parsing or app-specific decoding.

This paper addresses research question (RQ): How accurately can artifacts be automatically typed and categorized by type within a logical image? We focus on artifact type classification as an operationally meaningful first step: coarse categories (e.g., Image, Audio, Document, Log/Config, App Cache/Object) map directly to investigative actions (filtering, prioritization, and rapid scoping) and can be produced from lightweight signals available in nearly any logical acquisition.

We propose a lightweight pipeline that constructs features from file metadata, MIME/type hints, and path context signals, and trains supervised models for multi-class type prediction. To ensure realistic performance estimates in the presence of duplicates (e.g., cached copies and forwarded media), we evaluate using a SHA-256 grouped split that prevents duplicate leakage across train/validation/test [1-3]. Main contributions are as follows:

1) A triage-friendly artifact type classification pipeline for Android logical images using lightweight features.

2) A leakage-aware evaluation protocol based on SHA-256 grouped splitting for duplicated mobile datasets.

3) An empirical Android OS version 15 logical-image study demonstrating clear gains over a rule-based baseline and analyzing feature-group contributions via ablation.

The remainder of the paper is organized as follows: Section II reviews related work; Section III defines the problem; Sections IV–VI describe dataset, method, and experiments; Section VII discusses implications and limitations; and Section VIII concludes with future directions.

II. RELATED WORKS

A. Mobile DF guidance and acquisition constraints

Mobile device forensics is shaped by strict requirements on evidence integrity, reproducibility, and methodological transparency. NIST SP 800-101r1 frames mobile forensics procedures across acquisition, examination, analysis, and reporting, and highlights practical constraints that vary by device, OS, and extraction method [1]. Complementarily, ISO/IEC 27037:2012 provides guidance for identification, collection, acquisition, and preservation of digital evidence, supporting a "forensically sound" handling process [2]. These guidance documents motivate our focus on inventory-level triage for logical acquisitions, where deep parsing may be incomplete or unavailable but early decisions are still required.

B. Artifact extraction and validation frameworks

A dominant line of mobile DF work relies on artifact parsers that decode app/OS-specific storage formats into structured evidence (e.g., chats, locations, call records).

DOI: 10.25045/cyber.gov.2026.46

Brignoni's iLEAPP/ALEAPP emphasizes open-source parsing and, critically, the need for testing and validation of artifact extraction as the artifact surface continuously evolves [6,7].

Such frameworks are highly effective when parsers exist and the exported data contains the expected sources. However, in logical images, investigators frequently encounter large volumes of heterogeneous files (media, caches, partial exports) that are not immediately parsable, motivating complementary methods that operate directly on lightweight signals present in any file inventory.

C. Triage decision models and operational practice

Digital forensic triage has been studied both as an operational necessity and as a decision-making process requiring consistency and transparency. Horsman formalizes triage decision-making with staged models to support consistent practice when deciding whether and how to triage device contents under time and resource pressure.

This body of work underscores that triage is not merely a technical shortcut: it is a structured approach to managing investigative workload and uncertainty - precisely the context where inventory-level artifact typing can provide immediate value.

D. Metadata-based ML for smartphone file triage

The closest technical line to our approach uses file metadata as a low-cost proxy signal to support triage. Serhal and Le-Khac propose machine-learning approaches for smartphone file triage using features extracted from file metadata and compare multiple classifiers, showing that metadata-driven ML can effectively prioritize files for examination [4].

Our work differs in (i) targeting artifact type classification (coarse operational categories rather than "files of interest"), (ii) explicitly incorporating path-context signals alongside MIME/type hints and metadata, and (iii) enforcing a SHA-256 grouped split to control duplicate leakage, which is particularly relevant in mobile logical images where duplicates are common.

E. Android-specific triage tooling and scalable triage paradigms

Recent work explores practical triage tooling directly on Android devices, aiming to make selected data accessible quickly to investigators under constrained conditions. More broadly, triage paradigms such as bulk analysis (e.g., feature-driven scanning rather than full parsing) have been used to accelerate forensic processing in other digital media contexts, motivating our emphasis on lightweight, scalable signals for early-stage decisions.

F. Summary of gap addressed

Prior work establishes the importance of triage, the value of parser-based extraction, and the feasibility of metadata-driven ML. Yet, logical-image workflows still lack a robust, leakage-aware, inventory-level method for coarse artifact type classification that remains useful when deep parsing is incomplete [4]. Our study addresses this gap for an Android OS version 15 logical image by combining lightweight feature extraction with leakage-aware evaluation.

III. Dataset and Acquisition

A. Inventory Construction and Schema

The logical image is represented as a file inventory of N = 1331 files with total size 438.97 MB. Each record contains lightweight attributes extractable without deep content parsing:

- Paths: rel_path, abs_path
- Metadata: size_bytes, mtime_iso, ctime_iso (if available)
- Typing hints: ext, mime (and derived mime_major/subtype tokens when present)
- Context: bucket (operational folder grouping used as contextual signal)
- Quality flags: e.g., MIME–extension mismatch, symlink indicator

This schema is triage-oriented: all fields can be computed from file-system views and basic typing utilities, enabling scalable and reproducible processing across heterogeneous logical exports. The logical export was obtained using an ADB-based workflow, which provides file-system level access to user-accessible storage under common operational constraints and is therefore representative of practical first-pass mobile triage.

B. Labeling and Taxonomy Mapping

A labeled subset of n = 348 files was created for supervised training and evaluation under the coarse artifact-type taxonomy defined in Section IV-A. Labels are triage-oriented and intentionally coarse to support operational filtering. The labeled subset reflects realistic mobile storage behavior and is therefore class-imbalanced, with substantial mass in cache-like and media-related categories.

C. Label–Inventory Alignment and Hash Enrichment

To ensure consistent linkage between labels and inventory entries, the labeled table is merged back into the full inventory using (rel_path,bucket). Mobile logical images often contain duplicates (e.g., forwarded media, cached copies, thumbnails). To prevent optimistic estimates caused by duplicate leakage across splits, we adopt a SHA-256 grouped split where all files sharing the same hash are assigned to the same partition. The resulting split is: Train=251, Validation=28, and Test=69. Table 1 summarizes the dataset characteristics and leakage-aware split protocol used throughout the experiments

TABLE 1. DATASET SUMMARY AND LEAKAGE-AWARE SPLİT

№	*Item*	*Value*
1	Acquisition type	Android 15 logical image
2	Total files in inventory	1331

№	*Item*	*Value*
3	Total inventory size	438.97 MB
4	Labeled subset size	348
5	Label taxonomy	Coarse artifact types (triage-oriented)
6	Join keys (labels ↔ inventory)	(rel_path, bucket)
7	SHA-256 availability after merge	0.00% missing
8	Split protocol	SHA-256 grouped (leakage-aware)
9	Train / Validation / Test	251 / 28 / 69
10	Primary metric	macro-F1
11	Secondary metrics	weighted-F1, per-class PR/F1, confusion ma

IV. METHODOLOGY

A. Problem Formulation and Taxonomy

Let F denote the file inventory extracted from the Android logical acquisition. For ich $f \in F$, we compute lightweight features x_f (metadata, MIME/type hints, and path context) and predict a coarse artifact type $\hat{y}_f \in Y$

This study addresses targets of the RQ. The target is the inventory-level triage stage, avoiding deep content parsing (e.g., database decoding) and producing operational categories that support immediate analyst actions (filtering, scoping, prioritization) [3,5].

We define a triage-oriented taxonomy C: Image, Video, Audio, Document, Log/Config, Archive, Executable/Package, Database/Backup, App Cache/Object, and Unknown/Other. Separating App Cache/Object from Unknown/Other avoids forcing uncertain assignments while retaining a high-frequency, operationally meaningful triage class.

Given the inventory and a labeled subset, the pipeline proceeds as follows: (i) extract lightweight features from each file; (ii) train supervised classifiers with imbalance handling; (iii) predict artifact types under a leakage-aware split protocol; and (iv) evaluate performance using macro-F1/weighted-F1 with per-class diagnostics. The workflow Algorithm is summarized in Figure 1.

Input: Inventory F; labeled subset $L \subset F$; taxonomy C

Output: Predicted types $\{\hat{y}_i\}$

B. Lightweight Feature Extraction

We extract three groups of low-cost, parser-independent features:[6,7]

1) Metadata features. log(1+size_bytes), size bins, normalized extension token, and optional bucket context.

2) MIME/type-hint features. mime_major and subtype tokens when available.

3) Path-context features. Path depth, parent directory tokenization, and binary indicators for common forensic path keywords (e.g., dcim, camera, download, documents, cache, tmp, thumbnails, and app/vendor markers).

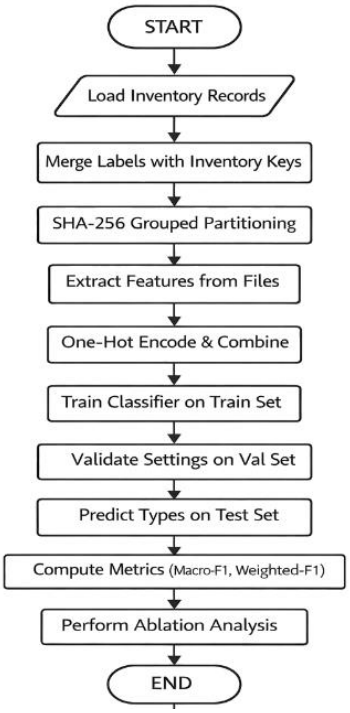

Figure 1. Lightweight metadata-and-path pipeline for artifact-type classification on Android logical images with duplicate-aware splitting

This representation is designed to remain robust under incomplete typing and heterogeneous app storage, while preserving triage speed.

C. Baseline and Learning Models

Heuristic baseline (Ext+MIME). A deterministic mapping from extension/MIME hints to artifact types with limited path-based disambiguation (e.g., .txt as Document vs. Log/Config based on context). This provides a transparent lower bound.

Logistic Regression. A class-weighted multinomial linear model serving as an efficient and interpretable baseline for sparse categorical indicators.

Random Forest (proposed). A balanced-subsample Random Forest to capture non-linear interactions (e.g., extension × path tokens; MIME × size bins) with minimal tuning.[8]

Categorical features are represented via one-hot encoding; numeric features are included directly.

D. Evaluation Metrics and Ablation Protocol

Given class imbalance typical of mobile logical inventories, we report macro-F1 as the primary metric and weighted-F1 and accuracy as secondary metrics. Macro-F1 reflects robustness across classes (including low-support types), while weighted-F1 captures operational utility under the observed distribution. We additionally report per-class precision/recall/F1 and confusion matrices for diagnostic interpretation. To quantify the contribution of lightweight cues, we perform an ablation study on the proposed RF model by removing one feature group at a time: Full (Metadata+MIME+Path), Path

(Metadata+MIME), MIME (Metadata+Path), and Metadata (MIME+Path), and evaluate all variants on the same SHA-256 hash-grouped split.

Reproducibility and artifacts. To support reproducibility, we provide the full implementation of the proposed pipeline, including feature extraction, duplicate-aware splitting, model training, and evaluation scripts. The repository also includes configuration files and step-by-step instructions to rerun the experiments end-to-end [10].

V. EXPERİMENTS AND RESULTS

A. Experimental Design and Reproducibility

This section evaluates RQ - the accuracy of automatic artifact detection and type classification within an Android OS version 15 logical image - under a protocol designed to be realistic for mobile logical inventories.

Data splits. All experiments use the SHA-256 grouped split described in Section III-E to prevent duplicate leakage (Train=251, Val=28, Test=69). This is critical because mobile logical images frequently contain duplicates due to caching, forwarding, thumbnails, and replicated media objects; random file-level splits would otherwise inflate generalization estimates.

Metrics. Given natural class imbalance in mobile inventories, we report macro-F1 as the primary metric and weighted-F1 as a secondary metric. Macro-F1 reflects robustness across classes (including minor classes), while weighted-F1 reflects operational utility under the observed distribution. We additionally report accuracy and per-class P/R/F1 for interpretability.

Model selection. Validation is used for minimal checks and sanity verification (e.g., confirming feature extraction consistency and preventing overfitting by ad-hoc tuning). In line with forensic reproducibility, we avoid extensive hyperparameter search and instead emphasize leakage-aware evaluation and interpretable baselines. Models were implemented using scikit-learn [9]. We compare three approaches aligned with practical triage workflows:

- Heuristic baseline (Ext+MIME rules): deterministic mapping from file extensions and MIME hints to coarse artifact types. This reflects common first-pass triage practice.
- Logistic Regression (LR): class-weighted multinomial LR trained on the full lightweight feature set; an interpretable linear baseline for sparse categorical indicators.
- Random Forest (RF, proposed): balanced-subsample RF trained on the same features; intended to capture non-linear interactions among metadata/MIME/path cues.

The primary outcomes are summarized in Table 2. Result interpretation. As shown in Table 2, learning-based models substantially outperform the rule baseline under leakage control, improving macro-F1 from 0.2496 to 0.5808 while achieving high weighted-F1 (0.9706) and accuracy (0.97) on the test split. These results directly answer RQ: coarse artifact types can be classified with meaningful triage accuracy in an Android logical image using lightweight, parser-independent signals, and the improvement over heuristic typing is substantial under a leakage-aware protocol.

TABLE 2. ARTİFACT TYPE CLASSİFİCATİON ON THE LEAKAGE-AWARE TEST SPLIT

Model	*Feature set*	*macro-F1*	*weighted-F1*	*Accuracy*
Heuristic (Ext+MIME rules)	Rules	0.2496	0.3001	0.39
Logistic Regression	Full	0.5808	0.9706	0.97
Random Forest (proposed)	Full	0.5808	0.9706	0.97

Table 2 reports the performance of the current lightweight classifier on the hash-grouped test split. The model reaches high overall accuracy and weighted-F1, indicating strong performance on the dominant classes in the dataset. However, the macro-F1 is substantially lower, which reflects degraded performance on minority classes under the same evaluation protocol. The main misclassifications occur between artifact categories that share similar file extensions and MIME types but differ in evidentiary context, where only subtle path cues are available. In addition, Android- and vendor-specific directory structures can introduce drift across versions and devices, which may reduce generalization to unseen environments. Finally, borderline cases can be sensitive to taxonomy operationalization, implying that some label noise is unavoidable and should be considered when interpreting macro-level metrics.

B. Contribution of Lightweight Feature Groups

To assess which feature groups drive performance, we ablate Path, MIME, and Metadata from the RF model (Section IV-E). Results are reported in Table 3.

TABLE 3. ABLATION RESULTS FOR RANDOM FOREST ON THE TEST SPLIT

Variant	*Included feature groups*	*macro-F1*	*weighted-F1*	*Accuracy*
RF (Full)	Metadata + MIME + Path	0.5808	0.9706	0.97
RF (–Path)	Metadata + MIME	0.5808	0.9706	0.97
RF (–MIME)	Metadata + Path	0.5490	0.9014	0.90
RF (–Metadata)	MIME + Path	0.5808	0.9706	0.97

Removing MIME/type hints causes the most notable degradation (weighted-F1 drops from 0.9706 to 0.9014; macro-F1 from 0.5808 to 0.5490), indicating that MIME signals carry substantial discriminative information for coarse typing in this logical-image setting. The ablation results for the proposed RF model are summarized in Table 3.

The lack of change for Path and Metadata should not be interpreted as "path/context is useless." Rather, under the current split and class distribution, MIME signals already

separate dominant classes very strongly; additional gains from path/metadata may become visible with broader device/app variation or better rare-class coverage (also consistent with the grouped split trade-off discussed in Threats to Validity).

C. *Per-Class Performance and Error Characterization*

Per-class precision, recall, and F1 for the proposed RF (full features) on the test split are reported in Table 4. Overall performance is strong for triage-critical categories present in the test set.

TABLE 4. PER-CLASS PRECİSİON, RECALL, AND F1 FOR THE PROPOSED RF (FULL FEATURES) ON THE TEST SPLİT.

Class	*Precision*	*Recall*	*F1*	*Support*
App Cache/Object	0.97	0.97	0.97	36
Audio	1.00	1.00	1.00	13
Database/Backup	1.00	1.00	1.00	2
Document	1.00	0.80	0.89	5
Image	0.90	1.00	0.95	9
Log/Config	1.00	1.00	1.00	4

As shown in Figure 2–3, the remaining errors concentrate in low-support classes and boundary cases where lightweight cues (MIME/type and path context) are similar across artifact types. This is consistent with the reduced recall observed for the Document class in Table 4.

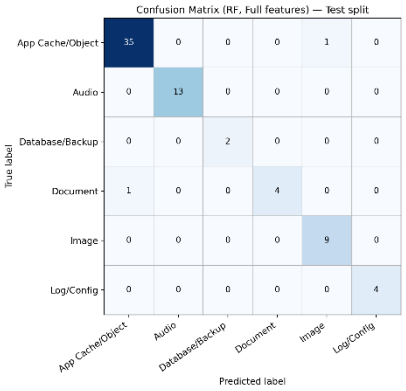

Figure 2. Confusion matrix (counts) of the proposed RF (full features) on the leakage-aware test split.

Table 4 indicates high utility for operational triage: dominant classes such as App Cache/Object and media categories achieve high F1, supporting rapid filtering of large inventories. The main residual weakness appears in Document recall (0.80), consistent with boundary cases where lightweight cues overlap (e.g., small text artifacts in app/cache-like paths).

Under leakage-aware evaluation on an Android logical image, lightweight signals enable effective coarse artifact typing. Relative to heuristic typing, supervised models deliver substantially higher macro-F1 while maintaining very high weighted-F1 and accuracy (Table 3).

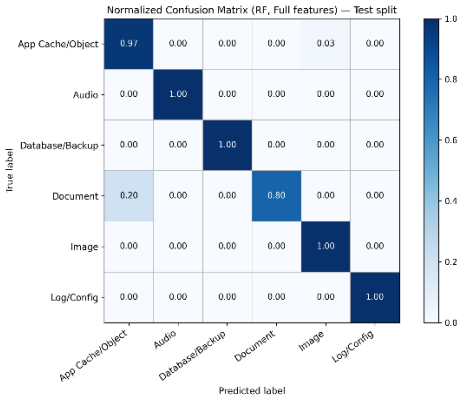

Figure 3. Normalized confusion matrix of the proposed RF (full features) highlighting boundary confusions between artifact types..

Error analysis. The main misclassifications occur between artifact categories that share similar file extensions and MIME types but differ in evidentiary context, where only subtle path cues are available. In addition, Android- and vendor-specific directory structures can introduce drift across versions and devices, which may reduce generalization to unseen environments. Finally, borderline cases can be sensitive to taxonomy operationalization, implying that some label noise is unavoidable and should be considered when interpreting macro-level metrics.

RQ asks how accurately heterogeneous files in an Android logical image can be classified by artifact type using lightweight cues (metadata and path signals). The obtained results show that high overall accuracy is achievable in this setting; however, macro-F1 indicates that minority artifact types remain challenging, motivating additional robustness measures and richer but still triage-friendly signals in future iterations.

VI. DİSCUSSİON

A. *Cyber Sovereignty and Digital Resilience Implications*

Evidence triage on mobile devices increasingly intersects with cyber sovereignty goals because investigations can involve sensitive citizen, governmental, or critical-sector data. The proposed on-premise, triage-friendly pipeline supports rapid prioritization without requiring cloud-based processing, thereby reducing external data exposure and supporting data sovereignty principles. From a digital resilience standpoint, faster artifact-type classification can shorten time-to-insight during incident response and forensic workflows, especially when mobile devices act as evidence carriers in incidents affecting critical systems and services. In this sense, the work aligns most directly with the conference themes on data sovereignty, applied AI solutions, and cybersecurity and digital resilience for critical systems..

B. *Summary of Key Findings and Answer to RQ*

This study assesses whether heterogeneous files in an Android logical image can be automatically typed with

sufficient accuracy to support inventory-level triage. Under a leakage-aware SHA-256 hash-grouped split, the proposed Random Forest classifier with lightweight cues achieves strong overall performance (macro-F1 = 0.5808, weighted-F1 = 0.9706, accuracy = 0.97), indicating that coarse artifact typing is practical and effective for first-pass inventory review. Therefore, RQ1 is answered positively: artifact type classification using metadata- and path-level signals can provide actionable triage value when duplicate leakage is controlled..

C. Practical Value for Mobile DF Triage

Operationally, the output of the pipeline functions as an immediate filtering and scoping layer: investigators can quickly focus on high-value categories (e.g., documents and logs/configs) while deprioritizing bulk cache-like objects during early review. This is particularly useful for Android logical exports where directory heterogeneity, mixed storage conventions, and incomplete typing signals can slow manual triage. Because the approach remains lightweight and requires no app-specific parsing, it is deployable early in DF workflows—before deeper artifact extraction, content parsing, or timeline reconstruction.

D. Why Heuristics Underperform and What Ablation Reveals

Rule-based typing from extensions and MIME hints is brittle in realistic mobile inventories: extensions may be missing or misleading, MIME hints can be coarse, and app storage introduces non-standard naming and containerization. Learning-based models integrate multiple weak cues (MIME tokens, metadata regimes, and contextual path markers), which improves robustness under noisy naming.

The ablation study (Table 3) indicates that MIME features are the primary driver of separability in the current test distribution: removing MIME substantially degrades performance (accuracy drops to 0.90 and weighted-F1 to 0.9014), whereas removing path or metadata yields minimal change relative to the full model. This pattern suggests that, for the dominant classes present in the evaluated logical image, MIME provides a strong discriminative signal; however, this does not imply that path or metadata are universally uninformative—rather, their contribution may become more apparent under broader device/app variability and richer rare-class coverage.

E. Error Characteristics and Class-Level Behavior

Macro-F1 and weighted-F1 should be interpreted jointly: weighted-F1 reflects strong operational utility on frequent classes, while macro-F1 provides a conservative view across classes under imbalance. The class-level report (Table 4) shows near-perfect performance for most categories but reduced recall for Document (recall = 0.80), which is particularly important for evidentiary prioritization. Confusion patterns (Fig. 2–3) are consistent with mobile storage behavior, where boundary cases (e.g., cache objects vs. user-facing media, or text-like artifacts stored within app folders) can share similar lightweight cues. Many such errors are triage-tolerant, but improving sensitivity for document- and log-like artifacts remains a key practical target. We note that several classes have limited support in the test split (Table 4), so class-level estimates should be interpreted cautiously.

VII. Threats to Validity

A. Internal Validity

A key threat is evaluation leakage caused by duplicated files (e.g., replicated media, cached copies, or backups) appearing across splits, which can inflate performance. We mitigate this risk using a SHA-256 hash-grouped splitting protocol, ensuring that identical file contents are confined to a single split. Another internal threat is unintended coupling between feature engineering and labels (e.g., dataset-specific path patterns). To reduce this, we restrict features to lightweight cues that are available at triage time and avoid any manual, case-specific rules in the learning pipeline.

B. Construct Validity

The notion of "artifact type" depends on the operational taxonomy and labeling guidelines. Borderline categories (e.g., system- vs application-related artifacts) may introduce label ambiguity and noise, which can disproportionately affect macro-level metrics. We partially address this by defining artifact types at a consistent granularity and reporting both macro-F1 and weighted-F1 to reflect minority-class sensitivity and overall support-weighted performance, respectively. Remaining ambiguity is discussed through error analysis and confusion patterns.

C. External Validity

Our experiments are conducted on a single Android OS version 15 logical image from a specific device and vendor environment, which limits generalization across devices, Android versions, and app storage conventions. Directory structures and naming patterns may drift across OS updates or OEM customizations, potentially degrading models that rely on path signals. While the current work targets a realistic triage setting, broader validation on multi-device and multi-version datasets is required before drawing strong general conclusions.

D. Conclusion validity

Results may be sensitive to the particular train/validation/test partitioning, especially under class imbalance and low-support categories. Although we use a duplicate-aware grouped split, a single split may still yield optimistic or pessimistic estimates. Future extensions will incorporate repeated hash-grouped splits (or grouped cross-validation) and report variability (e.g., mean ± standard deviation or confidence intervals) to better quantify statistical stability.

Conclusion and Future Work

This paper investigated RQ:—how accurately artifacts can be automatically typed within an Android OS version 15 logical image—using an inventory-level, lightweight feature pipeline that avoids deep content parsing. Under a SHA-256 hash-grouped (leakage-aware) split, supervised models substantially outperformed heuristic typing and achieved strong

triage performance (accuracy = 0.97, weighted-F1 = 0.9706), demonstrating that coarse artifact typing can be reliably automated in realistic logical-image conditions. The workflow is fast, parser-independent, and audit-friendly, making it suitable as an early-stage triage layer prior to deeper app-specific extraction.

Importantly, the results are directly relevant to conference priorities on data sovereignty, applied AI solutions, cybersecurity and digital resilience, and critical systems and infrastructures. Because the proposed approach can be deployed on-premise in institutional forensic laboratories, it supports sovereignty-aligned handling of sensitive evidence by reducing dependence on external cloud services and limiting data exposure during triage. In state institutions and critical-infrastructure investigations (e.g., energy and public services), this capability can shorten time-to-insight by rapidly scoping large logical exports, prioritizing document- and log-like artifacts, and deprioritizing bulk cache objects—thereby improving analyst throughput and strengthening operational resilience.

While the overall accuracy and weighted-F1 are high, macro-F1 is lower due to class imbalance: dominant classes contribute heavily to weighted metrics, whereas macro-F1 penalizes errors on minority classes equally. Fig. 2–3 indicate that remaining confusions are primarily boundary-case and low-support errors. In future works, to strengthen generalization, forensic defensibility, and operational deployment in sovereignty-sensitive environments, we will pursue:

1. Cross-device and cross-version validation. Evaluate across multiple vendors, Android versions, and app releases to quantify drift and portability beyond a single acquisition.
2. Stratified grouped evaluation with uncertainty. Use repeated hash-grouped splits and report variability (e.g., mean ± std or confidence intervals) to better characterize stability under imbalance and duplicate-aware partitioning.
3. Hierarchical typing (coarse→fine). Extend the taxonomy from coarse triage classes to finer-grained subtypes (e.g., thumbnails vs. user media; log families) while retaining interpretability.
4. Targeted lightweight cues. Add minimal file-signature/header checks and small content fingerprints (triage-safe) to reduce boundary confusions where MIME/extension is unreliable.
5. Workflow integration and runtime reporting. Integrate the pipeline into institutional DFIR workflows, report runtime/throughput metrics, and quantify investigator time savings on realistic case studies relevant to critical systems.
6. Reproducibility package. Provide a reproducible schema, feature extraction scripts, and the split protocol (with anonymization/surrogates when required) to enable benchmarking and peer comparison.

REFERENCES

[1] R. Ayers, S. Brothers, and W. Jansen, “Guidelines on Mobile Device Forensics,” NIST Special Publication 800-101 Revision 1, 2014.

[2] ISO/IEC, ISO/IEC 27037:2012—Information technology—Security techniques—Guidelines for identification, collection, acquisition and preservation of digital evidence, 2012.

[3] S. L. Garfinkel, “Digital media triage with bulk data analysis and bulk_extractor,” Computers & Security, vol. 32, pp. 56–72, 2013, doi: 10.1016/j.cose.2012.09.011

[4] C. Serhal and N.-A. Le-Khac, “Machine learning based approach to analyze file meta data for smart phone file triage,” Forensic Science International: Digital Investigation, vol. 37 (Supplement), Art. no. 301194, 2021, doi: 10.1016/j.fsidi.2021.301194.

[5] G. Horsman, “Triaging digital device content at-scene: Formalising the decision-making process,” Science & Justice, vol. 62, no. 1, pp. 86–93, 2022, doi: 10.1016/j.scijus.2021.12.001.

[6] A. Brignoni, ALEAPP (Android Logs, Events, and Protobuf Parser), GitHub repository. [Online]. Available: https://github.com/abrignoni/ALEAPP

[7] A. Brignoni, iLEAPP (iOS Logs, Events, and Plist Parser), GitHub repository. [Online]. Available: https://github.com/abrignoni/iLEAPP

[8] L. Breiman, “Random forests,” Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.

[9] F. Pedregosa et al., “Scikit-learn: Machine learning in Python,” Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011

[10] R. A. G. B., “artifact_classif_RQ1: Artifact type classification for Android logical images (RQ1 pipeline),” GitHub repository. Accessed: 2026-02-15. [Online]. Available: https://github.com/rahibagb/artifact_classif_RQ1