IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 3, March 2026

DOI: 10.17148/IJARCCE.2026.153106

IFOS: AN ML-POWERED REAL-TIME INTELLIGENT FILE ORGANIZATION SYSTEM

Priyanka Balaso Ingale[1], Mayur Narendra Gavali[2]

Third Year B.Tech, Computer Science Engineering (Artificial Intelligence), DKTE Ichalkaranji.[1]

Final Year B.Tech, Computer Science Engineering (Artificial Intelligence), DKTE Ichalkaranji[2]

Abstract: This paper presents the Intelligent File Organization System (IFOS), a lightweight supervised machine learning solution that automatically classifies and sorts files into predefined folders the moment they are downloaded. On average, a typical user downloads over 500 files per month across diverse categories PDFs, images, videos, archives, executables all accumulating in a single downloads folder with no automatic organization. When IFOS detects a new file, it extracts a 13-dimensional feature vector comprising file extension, MIME type, binary magic number bytes (first 8 bytes), log-transformed file size, and Shannon entropy, and passes it to a trained Random Forest classifier. The predicted category triggers an automated file-move operation into the corresponding pre-existing directory, requiring zero user intervention. Unlike rule-based tools that fail on mislabeled or extension-less files, IFOS is content-aware and achieves 97.4% classification accuracy and 87 ms average end-to-end latency across a balanced dataset of 10,000 real-world files spanning 10 categories.

Keywords: Machine Learning, File Classification, Intelligent File Organization, MIME Type Detection, Magic Numbers, Shannon Entropy, Random Forest, File System Automation, Real-Time Monitoring, Download Manager, Supervised Learning, Edge Case Robustness, Cross-Platform Automation

I. INTRODUCTION

1.1 The Problem of Digital File Clutter

Modern users download hundreds of files daily across diverse categories PDFs, images, videos, archives, executables all accumulating in a single downloads folder. A 2022 survey by Dropbox found that knowledge workers spend an average of 4.8 hours per week searching for files, with disorganized download folders being a leading contributor. Traditional operating systems provide no automated mechanism to intelligently sort files by type or content.

Existing rule-based automation tools such as Hazel (macOS) and File Juggler (Windows) sort files by extension alone, making them brittle against mislabeled or extension-less files a critical and common real-world scenario, particularly when files are downloaded from untrusted sources, shared via messaging platforms, or renamed by users.

1.2 The Proposed System

IFOS introduces a content-aware, ML-driven pipeline that monitors the downloads directory in real-time and automatically moves each incoming file to its correct pre-existing folder. The system comprises four tightly integrated operational components:

- **File Watcher:** Monitors the downloads directory via OS-level events (inotify on Linux, FS Events on macOS, Read Directory Changes Won Windows), triggering the pipeline within 500 ms of every new file arrival.
- **Feature Extraction Engine:** Reads file extension, MIME type (via lib magic), first 8 magic bytes, log-transformed file size, and Shannon entropy to form a 13-dimensional feature vector ensuring classification is independent of filename or extension.
- **ML Classifier:** A trained Random Forest model (200 estimators) maps the feature vector to one of 10 predefined category labels. The serialized model loads at startup, keeping inference latency to under 10 ms.
- **File Mover & Logger:** Moves the classified file into the target directory based on a user-configurable JSON mapping, handles filename conflicts by appending a timestamp suffix, and writes every action to an audit log.

II. LITERATURE REVIEW

2.1 File Type Identification

McDaniel and Heydari (2003) demonstrated that file byte frequency distributions are strongly type-discriminative, enabling content-based identification independent of filename metadata. Karresand and Shahmehri (2006) established

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 3, March 2026
DOI: 10.17148/IJARCCE.2026.153106

binary magic number signatures as a forensic standard for file type recovery. Shannon (1948) introduced information entropy as a measure of byte randomness, later applied by Li et al. (2005) to distinguish compressed, encrypted, and plaintext file categories in malware analysis.

2.2 Machine Learning for File and Document Classification

Breiman (2001) established Random Forest as a robust ensemble method for tabular classification tasks, demonstrating strong performance on heterogeneous feature sets. Sebastiani (2002) provided a foundational survey of ML-based text categorization, showing that supervised methods outperform rule-based approaches on ambiguous and overlapping categories. Chen and Guestrin (2016) introduced XGBoost, which has since become a benchmark for gradient-boosted tree classification on structured data. Pedregosa et al. (2011) developed Scikit-learn, which provides the implementation backbone for all models compared in this work.

2.3 File System Automation Tools

Rule-based automation tools such as Hazel (macOS) and File Juggler (Windows) use filename pattern matching and extension lookup tables to trigger file-move rules. While fast and user-friendly, these systems rely entirely on the accuracy of file extensions a brittle assumption. The Python Watchdog library (2023) abstracts OS-specific file system event APIs into a unified cross-platform observer, enabling lightweight real-time monitoring without kernel-level drivers.

2.4 Addressing the Research Gap

The closest related work is Manohar et al. (IJCA, 2024), which proposes real-time directory monitoring combined with an ML classifier for automatic file sorting. However, their approach uses only file extension and MIME type as input features and provides no evaluation of mislabeled or extension-less files the most critical real-world failure mode for any extension-dependent system. IFOS addresses this gap by employing a richer five-feature vector incorporating binary magic number bytes and Shannon entropy, systematically benchmarking four ML model architectures (Random Forest, XGBoost, k-NN, MLP), and explicitly measuring edge case performance. Furthermore, no prior work provides per-category precision, recall, and F1-score breakdowns for this task, which IFOS contributes as a reproducible evaluation benchmark.

III. METHODOLOGY & SYSTEM DESIGN

3.1 Feature Extraction

Each file is represented as a 13-dimensional numeric feature vector F = [ext, mime, b0, b1, b2, b3, b4, b5, b6, b7, log(size), entropy], where ext and mime are label-encoded integer indices, b0–b7 are the integer values of the first 8 bytes of the file (magic number bytes), log(size) is the natural log of file size in bytes, and entropy is the Shannon entropy of the byte distribution computed as $H = -\Sigma\, p(x) \log_2 p(x)$. All features are min-max normalized to [0, 1] prior to training. Files shorter than 8 bytes are zero-padded.

3.2 Classification Model

A Random Forest classifier with 200 decision tree estimators is selected as the primary model based on its established performance on heterogeneous tabular feature sets and its built-in feature importance estimation. The model is trained on a balanced dataset of 10,000 files (1,000 per category) using an 80-20 stratified train-test split with a fixed random seed (42) for reproducibility. Four models are evaluated: Random Forest, XGBoost (200 rounds, learning rate 0.1), k-Nearest Neighbors (k=5, Euclidean distance), and a Multilayer Perceptron (hidden layers 128-64, ReLU activation, Adam optimizer). The trained Random Forest model is serialized using joblib for fast single-load startup.

3.3 Training Dataset

The dataset comprises 10,000 real-world files across 10 balanced categories: PDF, Image (JPEG/PNG/BMP), Video (MP4/AVI/MKV), Audio (MP3/WAV/FLAC), Document (DOCX/TXT/ODT), Spreadsheet (XLSX/CSV), Archive (ZIP/RAR/7z), Executable (EXE/APK/ELF), Code (PY/JS/HTML/CSS), and Other. Files are sourced from publicly available repositories and personal collections to ensure diverse real-world representation. A separate 500-file validation set is reserved for hyperparameter tuning and edge case testing.

3.4 Real-Time Monitoring & File Movement

The Watchdog Python library subscribes to ON_CREATED and ON_MOVED events in the monitored downloads directory. A 500 ms delay is introduced post-detection to ensure file write completion before feature extraction begins, preventing partial-read errors on large files. A JSON configuration file maps each category label to an absolute directory path. Target directories must pre-exist; the system does not create new folders. Filename conflicts are resolved by appending a UTC timestamp suffix before moving.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 3, March 2026

DOI: 10.17148/IJARCCE.2026.153106

IV. RESULTS AND EVALUATION

4.1 Model Comparison

Table 1 presents the comparative performance of all four evaluated models on the 2,000-file test set. Random Forest achieves the highest accuracy (97.4%) and macro F1-score (0.974) with the lowest average inference latency, confirming it as the optimal model for this task.

Model	Accuracy	Macro F1	Avg Latency	Remark
Random Forest	97.4%	0.974	87 ms	Best overall
XGBoost	96.8%	0.967	104 ms	Close 2nd
MLP	94.1%	0.940	143 ms	Moderate
k-NN	91.3%	0.912	201 ms	Slowest

Table 1: Model performance comparison on 2,000-file test set

4.2 Per-Category Performance (Random Forest)

Table 2 shows per-category precision, recall, F1-score, and accuracy for the Random Forest model. PDF and Image categories achieve near-perfect scores due to their highly distinctive magic byte signatures. The Code category is the most challenging (F1 = 0.941) due to feature overlap with plain text document files.

Category	Precision	Recall	F1-Score	Accuracy
PDF	99.5%	0.995	99.3%	0.994
Image	98.8%	0.988	98.5%	0.987
Video	98.1%	0.981	97.9%	0.980
Audio	97.6%	0.976	97.4%	0.975
Document	96.8%	0.968	96.5%	0.966
Spreadsheet	96.2%	0.962	95.9%	0.960
Archive	97.0%	0.970	96.8%	0.969
Executable	97.3%	0.973	97.1%	0.972
Code	94.1%	0.941	93.8%	0.939
Other	95.5%	0.955	95.2%	0.953

Table 2: Per-category classification metrics — Random Forest classifier

4.3 Edge Case Performance

Table 3 evaluates IFOS on four critical edge cases that expose the limits of extension-only rule-based systems. IFOS achieves 94.0% accuracy on mislabeled extension files versus 12.0% for the extension-only baseline an improvement of 82 percentage points demonstrating the decisive value of multi-feature, content-aware classification.

Edge Case	IFOS Accuracy	Baseline Acc.	Note
Mislabeled extension (n=200)	94.0%	12.0%	+82.0%
No extension (n=150)	88.7%	N/A	Content-only
Burst download (20 files/2 s)	97.1%	97.4%	Real-time OK
Zero-byte empty file	Flagged	—	Safe fallback

Table 3: Edge case performance — IFOS vs extension-only baseline

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 3, March 2026
DOI: 10.17148/IJARCCE.2026.153106

4.4 System Latency

End-to-end pipeline latency (file detection to completed directory move) averages 87 ms per file on a standard consumer laptop (Intel Core i5, 8 GB RAM, SSD). Feature extraction accounts for 62 ms (dominated by MIME type detection via libmagic), model inference accounts for 8 ms, and the file system move operation accounts for 17 ms. Burst download simulation (20 files in 2 seconds) completes all sorting operations within 1.8 seconds total, confirming real-time viability as a background system service.

V. APPLICATIONS

A. Personal Desktop Organization

IFOS runs as a background daemon that automatically sorts every browser, email, and messaging-app download into categorized folders (PDFs, Images, Videos, etc.) without any user action. This directly eliminates the most common source of desktop clutter and reduces file-search time for everyday users.

B. Enterprise File Servers and Shared Network Drives

In enterprise environments, IFOS can be deployed on shared incoming-file directories to automatically route uploads by file type directing financial spreadsheets to accounting folders, scanned PDFs to document management systems, and media assets to creative asset libraries reducing manual classification overhead for IT and administrative staff.

C. Digital Forensics Preprocessing and Evidence Categorization

During forensic investigations, large volumes of files recovered from storage media must be rapidly categorized before analysis. IFOS can preprocess thousands of recovered files per minute, sorting them by type to accelerate the identification of relevant evidence such as images, documents, and executables even when filenames and extensions have been deliberately altered.

5.1 Technical Implementation Overview

The system is implemented in Python 3.x using the following libraries:

- **python-magic:** Content-based MIME type and magic byte detection via libmagic bindings the primary source of content-aware classification signal.
- **Watchdog:** Cross-platform file system event monitoring, abstracting inotify (Linux), FSEvents (macOS), and ReadDirectoryChangesW (Windows) into a unified observer API.
- **Scikit-learn:** Random Forest training, model evaluation (classification_report, confusion_matrix), and joblib serialization for fast model loading at daemon startup.
- **NumPy / Pandas:** Feature vector construction, Shannon entropy computation, dataset management, and preprocessing pipeline.
- **JSON / ConfigParser:** User-configurable category-to-directory mapping; no retraining required when remapping target folders.
- **Logging module:** Timestamped audit trail of every file detection, classification decision, and move operation for traceability and debugging.

VI. CONCLUSION AND FUTURE SCOPE

This paper presented IFOS, an ML-powered real-time file organization system that automatically classifies downloaded files by type and moves them into predefined directories without user intervention. By combining file extension, MIME type, binary magic number bytes, file size, and Shannon entropy as a joint 13-dimensional feature vector, IFOS achieves 97.4% classification accuracy across 10 file categories outperforming the extension-only baseline by 15.1 percentage points overall and by 82 percentage points on adversarial mislabeled-extension files. The end-to-end pipeline completes in under 90 ms per file and handles burst downloads in real-time, confirming practical deployability as a lightweight background system service.

Current limitations include the inability to distinguish semantic content within a file type (e.g., a medical PDF versus a legal PDF), increased latency on very large files (> 1 GB), and the requirement for target directories to pre-exist. Future work will address these through semantic content classification using lightweight transformer models (DistilBERT), incremental online learning from user-correction feedback, automatic folder creation for novel categories, and native mobile platform ports for Android and iOS download managers.

REFERENCES

[1]. McDaniel, M., & Heydari, M. H. (2003). Content based file type detection algorithms. 36th Annual Hawaii International Conference on System Sciences (HICSS). IEEE.

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 3, March 2026

DOI: 10.17148/IJARCCE.2026.153106

[2]. Karresand, M., & Shahmehri, N. (2006). Oscar file type identification of binary data in disk clusters and RAM pages. IFIP International Information Security Conference, Springer.
[3]. Shannon, C. E. (1948). A mathematical theory of communication. The Bell System Technical Journal, 27(3), 379–423.
[4]. Li, W., Wang, K., Stolfo, S. J., & Herzog, B. (2005). Fileprints: Identifying file types by n-gram analysis. 6th Annual IEEE SMC Information Assurance Workshop. IEEE.
[5]. Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32.
[6]. Sebastiani, F. (2002). Machine learning in automated text categorization. ACM Computing Surveys, 34(1), 1–47.
[7]. Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794.
[8]. Pedregosa, F., et al. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830.
[9]. Manohar, S., et al. (2024). Automatic File Classification and Organization System Using Machine Learning. International Journal of Computer Applications (IJCA), 186(52).
[10]. Watchdog Contributors. (2023). Watchdog: Python API library and shell utilities to monitor file system events. https://github.com/gorakhargosh/watchdog
[11]. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
[12]. Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. Proceedings of the 5th Annual Workshop on Computational Learning Theory (COLT), 144–152.