International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 13 Issue: 11 | Nov 2026 www.irjet.net p-ISSN: 2395-0072

Static PE-Header Features for Malware-versus-Benign Windows Executable Classification: A Reproducible Baseline and Feature-Reduction Study

Naitik Kambre[1], Sanket Hake[2], Rohit Kadam[3]

[123]*Students, Master of Computer Applications, D Y Patil Schools of Humanities & Sciences, Navi Mumbai, India*

Abstract - *Static Portable Executable (PE) header features are a low-cost candidate for a first-stage screen against malware, but reported results for such features mostly come from single random splits of curated data. This note evaluates 15 static features (14 PE-header fields and one content-derived indicator) for classifying Windows executables as malware (positive class) or benign, using a 62,485-sample CSV obtained from Kaggle. The label encoding (0 = malicious, 1 = benign) and the dataset's provenance are documented in secondary sources but not independently verified. The positive class consists of VirusShare-named samples, and the file gives no family labels. Across three repetitions of stratified five-fold cross-validation (15 held-out folds), a 100-tree Random Forest (RF) reached 99.677 ± 0.058% accuracy and 99.947 ± 0.026% ROC-AUC, against 86.442 ± 0.400% accuracy for standardized Logistic Regression (LR). With feature selection inside each training fold, the top-10 RF, which excludes the BitcoinAddresses indicator, reached 99.637% accuracy; the top-5 and top-3 RFs reached 98.843% and 97.491%. The RF–top-10 difference is not statistically resolved (corrected resampled t-test, p = 0.21). A direct ablation without BitcoinAddresses changes accuracy by 0.009 points (p = 0.54). These results measure within-dataset separability under random folds. They do not establish detection of unseen or evolving malware. On an external ransomware dataset, the top-10 model's accuracy fell to 68.29% (ROC-AUC 77.92%), so the in-dataset result does not transfer. The experiment script is provided for replication.*

Keywords - malware detection, Portable Executable, static analysis, Random Forest, feature selection, reproducibility

I. INTRODUCTION

Static analysis inspects an executable without running it, and fields of the Windows Portable Executable (PE) header are a common source of features [1]. Classifiers built on such fields, including Random Forest (RF) and Logistic Regression (LR), have reported high accuracy on curated datasets and on reduced feature sets [2]. Ransomware, whose behaviour (file encryption, ransom demands) has mostly been studied through dynamic or behavioural analysis [3], [4], is one threat class for which a cheap header-only first-stage screen is attractive. Its value, however, depends on label quality, sample diversity and evaluation design [5].

Scope. This paper is a malware-versus-benign study, not a family-level or ransomware-specific study. The supplied dataset is a public Kaggle CSV whose feature schema matches that of the MLRD ransomware-detection project [6]. The task is treated as binary malware detection with the malicious samples as the positive class. The study makes no claim of detecting ransomware specifically, attributing samples to families, or outperforming deployed products.

We ask: **RQ1** how do RF and LR compare under identical folds? **RQ2** how much performance does RF retain with the top 10, 5 or 3 features selected inside each training fold? **RQ3** does performance depend on the content-derived BitcoinAddresses indicator? **RQ4** are the RF results stable across different fold assignments? Contributions are (i) explicit documentation of dataset provenance and label-mapping assumptions, (ii) a leakage-aware feature-reduction evaluation repeated over three fold assignments, (iii) an analysis of BitcoinAddresses, (iv) an external test on a second ransomware dataset that is only partly independent, and (v) a released analysis script.

II. RELATED WORK

PE-header studies followed: Rezaei and Hamze [2] reported 95.59% accuracy with nine selected header features (as reported by the original authors, under their own protocols).

Larger benchmarks shifted attention to richer static representations. EMBER [7] released static features for over one million PE files. These benchmarks address malware broadly; a header-only study on malware samples without family labels is narrower and is positioned here as a baseline.

Ransomware detection has relied mostly on behaviour: Kharraz et al. [3] analysed ransomware activity across families, and Berrueta et al. [4] survey such techniques. Static screening complements them only if it generalises, which random splits cannot show.

Evaluation practice. Arp et al. [5] list pitfalls including sampling bias, label inaccuracy and data snooping, and TESSERACT [8] shows that ignoring temporal and spatial bias inflates reported performance. Feature selection performed outside cross-validation biases estimates [9]; this study therefore selects features inside each training fold and

states plainly that random folds cannot test temporal generalisation.

III. DATASET

A. Source and label mapping

The supplied file was obtained from Kaggle [10] and contains 62,485 rows and 18 columns: FileName, md5Hash, 15 numeric features (14 PE-header fields and the BitcoinAddresses indicator) and a Benign label. The feature schema matches that of the MLRD project [6], whose public repository describes PE-header extraction and YARA-based checks for embedded Bitcoin addresses. We adopt the Benign = 1 mapping for benign files. The file does not confirm the ransomware reading of Benign = 0, so this paper uses the neutral term malicious.

The mapping is documented in secondary sources but not verified against the binaries. The Kaggle dataset page, as retrieved on 31 October 2026, gives a one-line description, no label definitions, collection dates or row count, and lists the licence as Apache 2.0. The MLRD repository's data_file.csv [6] has the same 18-column header and the same counts as the supplied file (62,485 rows; 35,367 with Benign = 0; 27,118 with Benign = 1). Its filenames show what the labels rest on. Every Benign = 0 row is named VirusShare_ followed by its md5Hash, and no Benign = 1 row has that form. The Benign = 1 rows carry original Windows file names, such as calc.exe, firefox.exe and ONENOTE.EXE. The positive class is therefore a set of samples from the VirusShare malware collection. The file gives no family labels, so nothing in it shows that these samples are ransomware. As a limited sanity check, the two rows whose filenames match well-known benign programs (calc.exe and firefox.exe) carry Benign = 1, which is consistent with the mapping; the full-file check below is the stronger evidence. The file itself carries no documentation, so its exact collection dates, per-source counts and labelling procedure are unconfirmed. Because the file combines benign and VirusShare samples, labels may reflect collection source rather than independent expert verdicts. Confirming the Kaggle dataset's documentation remains outstanding. A stratified random sample of md5Hash values has not been checked against a public reputation service; this is stated as a limitation in Section VII.

B. Scope and BitcoinAddresses

BitcoinAddresses is not a header field: in MLRD it is the outcome of a YARA content rule that detects embedded Bitcoin addresses [6]. The feature set is thus "PE-header-style plus one content-derived indicator". Although payment addresses are plausible in ransomware, the feature name is not a validated family label and the indicator is more common in benign rows than in Benign = 0 rows: 714 of the 27,118 Benign = 1 rows (2.63%) and 411 of the 35,367 Benign = 0 rows (1.16%) have BitcoinAddresses = 1, including OneDriveStandaloneUpdater.exe and ONENOTE.EXE. Our evidence on its role is indirect but informative: BitcoinAddresses has a mean importance of 0.08% (Table III, rank 15 of 15), is outside the top 10 in all 15 folds, and the top-10 model loses 0.040 accuracy points relative to the full model. A direct ablation without the feature changes accuracy by 0.009 points ($p = 0.54$; Section V), so accuracy does not appear to depend on it.

C. Audit and preprocessing

All 62,485 rows have all 18 fields, with no empty cells, no duplicate FileName values and no duplicate MD5 hashes. Label 0 (malicious) has 35,367 rows (56.6%) and label 1 (benign) 27,118 (43.4%); the data are moderately imbalanced. FileName and md5Hash are excluded from the model inputs. The field is referred to as IatVRA throughout, matching the CSV column header, and corresponds to the IAT RVA field.

IV. EXPERIMENTAL DESIGN

Stratified five-fold cross-validation (shuffled) was repeated with three random seeds (42, 43 and 44), giving 15 held-out evaluation folds. Within each repetition, all models use identical folds. RF [11] used 100 trees with random_state set to the repetition seed; LR used StandardScaler followed by LogisticRegression (max_iter = 2000), both in scikit-learn [12]. No hyperparameter search was performed. For feature reduction, a 15-feature RF was fitted on each training fold and features were ranked by impurity importance computed on that training fold only, which avoids selection bias [9]; separate RFs were then fitted on the top 10, 5 and 3 features. For each fold, the 15-feature RF used for ranking is the same fitted model reported as the full-feature RF for that fold. Impurity importance can favour some feature types [13] and is not read causally.

The malicious class (Benign = 0) is the positive class. Fold-level metrics (accuracy, precision, recall, F1, ROC-AUC) are reported as mean ± sample SD over the 15 folds. Because the classes are moderately imbalanced, precision and recall are reported alongside ROC-AUC, since precision-recall analysis is more informative than ROC analysis on imbalanced data [14]; a PR-AUC or average-precision summary is not reported in this version. For the full RF we also report, for seed 42, the pooled out-of-fold confusion matrix, false-positive and false-negative rates with 95% Wilson intervals, balanced accuracy and the Matthews correlation coefficient (MCC) [15], which remains informative under class imbalance. Fold estimates are correlated, so the SD across folds understates uncertainty in model differences [16]. We therefore apply the corrected resampled t-test of Nadeau and Bengio [16] to per-fold accuracy differences ($k = 15$ folds, test-to-training ratio 1/4). This correction was derived for repeated random train/test splits; the folds here come

from three repeated k-fold runs whose test sets overlap across repetitions, so the test is an approximation. Four model comparisons are reported without adjustment for multiple comparisons. Differences that are not significant at $p < 0.05$ are treated as unresolved, and no further significance claims are made.

V. RESULTS

Table I summarises the repeated evaluation. RF with all 15 features is best on every metric, including recall. The gap to LR (13.2 points of accuracy) is far larger than the fold SD of either model (0.058 for RF and 0.400 for LR). LR's weakness is precision (84.0%): roughly one in six of its malware alerts is a false alarm. The top-10 RF is 0.040 points below the full model, a difference the corrected resampled t-test does not resolve ($t = 1.33$, $p = 0.21$). Top-5 and top-3 lose 0.834 and 2.186 points respectively; both differences are significant under the same test ($p < 0.001$). For reference, a majority-class predictor would score 56.6%.

TABLE I. Mean ± SD over 15 held-out folds (three repetitions of stratified five-fold CV, seeds 42–44) (%); malicious samples

Model	Accuracy	Precision	Recall	F1	ROC-AUC
RF (15 features)	99.677 ± 0.058	99.643 ± 0.093	99.786 ± 0.050	99.715 ± 0.051	99.947 ± 0.026
RF (top 10)	99.637 ± 0.035	99.622 ± 0.058	99.738 ± 0.052	99.680 ± 0.031	99.934 ± 0.031
RF (top 5)	98.843 ± 0.265	98.791 ± 0.251	99.170 ± 0.234	98.980 ± 0.233	99.736 ± 0.101
RF (top 3)	97.491 ± 0.138	97.125 ± 0.193	98.483 ± 0.300	97.799 ± 0.123	98.977 ± 0.224

Feature selection is performed inside each training fold. Top-k models use the k features with highest impurity importance on the training fold.

The paper's Section V re-implementation counts (129 false positives and 72 false negatives, 99.678%) did not reproduce. Re-running the released protocol gives 126 false positives and 79 false negatives for seed 42 (99.672%), which matches the original single run. Repeating seed 42 gave identical predictions. Across the three seeds, false positives range from 121 to 132 and false negatives from 72 to 79. The cause of the earlier re-implementation difference is not identified, because the original environment was not recorded. Table II reports the reproduced counts for seed 42.

TABLE II. Pooled out-of-fold confusion matrix, 15-feature RF, seed 42.

Actual class	Pred. benign	Pred. malicious	Total
Benign (label 1)	26,992	126	27,118
Malicious (label 0)	79	35,288	35,367
Total	27,071	35,414	62,485

Seed 42. Malicious samples (Benign = 0) are the positive class. FPR 0.465% (95% CI 0.390–0.553%); FNR 0.223% (95% CI 0.179–0.278%); balanced accuracy 99.656%; MCC 0.993.

Table II gives the pooled out-of-fold result for the full RF with seed 42: 126 benign files were flagged as malicious (false-positive rate 0.465%, 95% CI 0.390–0.553%) and 79 malicious files were missed (false-negative rate 0.223%, 95% CI 0.179–0.278%). Balanced accuracy is 99.656% and MCC is 0.993. Seeds 43 and 44 give balanced accuracy of 99.649% and 99.675% and MCC of 0.993 and 0.994.

Table III lists mean impurity importances over the five training folds for seed 42. DllCharacteristics ranks first (21.67%), followed by DebugRVA, DebugSize and MajorLinkerVersion. These are relative scores within the fitted forests; they are neither the percentage of malware detected nor evidence of attacker intent.

TABLE III. Top ten features by mean impurity importance over the five seed-42 training folds (all 15 features sum to 100%).

Rank	Feature	Mean importance (%)
1	DllCharacteristics	21.67
2	DebugRVA	12.81
3	DebugSize	12.74
4	MajorLinkerVersion	11.83
5	ResourceSize	8.30
6	MajorOSVersion	7.92
7	Machine	4.95
8	NumberOfSections	4.27
9	SizeOfStackReserve	4.02
10	IatVRA	3.84

BitcoinAddresses has mean importance 0.08% (rank 15 of 15, seed 42 only).

External validation. The Mendeley Ransomware PE Header Feature Dataset [17] contains 2,157 Windows executables:

1,023 ransomware from 25 families and 1,134 goodware. The dataset page asks users to cite the authors' 2023 article in Computers & Security [18]. Its features are the first 1,024 bytes of each PE header. The 14 header fields can be re-derived from those bytes, but BitcoinAddresses cannot, because it is a content-derived indicator. The top-10 features were chosen on the full MLRD data, using the same RF settings and seed 42, and are the same 10 fields as Table III. A top-10 RF was trained on all MLRD rows and applied unchanged to the external set, with ransomware as the positive class. Table IV reports the results.

TABLE IV. External validation on the Mendeley ransomware dataset [17], 2,157 files (%); ransomware is the positive class.

Model	Accuracy	Precision	Recall	Specificity	ROC-AUC
RF, top 10 (trained on MLRD)	68.29	75.96	48.49	86.16	77.92
RF, 14 header features	66.85	71.63	49.85	82.19	81.50

On the external set, accuracy fell from 99.64% in-dataset to 68.29%, and ROC-AUC fell from 99.93% to 77.92%. The 14-feature model gave 66.85% accuracy and 81.50% ROC-AUC. The majority-class baseline is 52.57%. Detection ranged from 0% to 100% across the 25 ransomware families, and 86.16% of goodware were classified correctly.

The medians show the shift. DebugRVA is 0 for MLRD malicious samples but 7,696 for external ransomware, and ResourceSize is 4,096 against 0. Thresholds learned on MLRD therefore do not transfer, which is consistent with the collection-source explanation in Section VI. Two limits apply. The features were re-derived from header bytes, so they may differ from the MLRD extractor's definitions, and that extractor was not inspected. The ransomware samples partly share VirusShare as a source with the MLRD positives, so the test is only partly independent.

VI. DISCUSSION

Within this dataset, RF separates the classes almost perfectly under random stratified folds, and the result holds across three fold assignments (RQ4). The large RF–LR gap suggests non-linear decision boundaries. For RQ3, removing BitcoinAddresses changes accuracy by 0.009 points ($p = 0.54$), so there is no detectable dependence on the feature; the top-10 comparison (0.040 points) agrees. More notable is that three features reach 97.5% accuracy and that the leading four (DllCharacteristics, DebugRVA, DebugSize, MajorLinkerVersion) describe security flags, debug data and linker settings. Because the two sample sets differ in source (Section III-A), we hypothesise that such fields capture differences in build toolchain or collection source between the two sample sets rather than malware behaviour; the external test is consistent with this (Section V), and the same effect would explain high random-split scores that fail on external data [5], [8]. A compact model may reduce feature-extraction work, but latency and memory were not measured, so no efficiency claim is made. Header screening is best viewed as one layer of a layered defence [1].

VII. LIMITATIONS AND FUTURE WORK

(1) Provenance and labels remain unverified (Section III-A): the positive class consists of VirusShare-named samples, the file gives no family labels, and the Kaggle page gives no label definitions, so ransomware status is not established. The source documentation remains to be confirmed.

(2) The CSV has no dates or family labels, so temporal drift, family overlap and source leakage are untested; random folds measure within-dataset generalisation only.

(3) The classes are moderately imbalanced.

(4) Importance scores are model-dependent and non-causal.

(5) External validation on one ransomware dataset [17] (Section V) is only partly independent, because its ransomware samples also come from VirusShare, and it covers 2,157 files; the repeated cross-validation addresses sensitivity to fold assignment only.

(6) Adversarial robustness, run-time cost and deployment behaviour were not measured.

(7) Only RF and LR were trained and compared; the majority-class rate is reported only as a reference figure, and gradient-boosting and other trained baselines and grouped validation are not reported in this version.

(8) The cause of the difference between the earlier and present seed-42 confusion-matrix counts has not been identified (Section V).

(9) Every positive sample is named VirusShare_<md5> and every benign sample carries an original Windows file name, so labels are confounded with collection source; the high accuracy may reflect that difference rather than malware behaviour.

(10) A stratified random sample of md5Hash values has not been checked against a public reputation service.

Future work should use an independently collected, family-labelled malware dataset with PE-header features, time-based and family-grouped splits, calibration and class-specific error analysis, and robustness tests against header manipulation.

VIII. CONCLUSION

A stratified five-fold evaluation of 15 PE-header-style features on a Kaggle dataset whose positive class consists of VirusShare-named malware samples, repeated over three fold assignments, found that a 100-tree Random Forest achieved 99.677 ± 0.058% accuracy and 99.947 ± 0.026% ROC-AUC (MCC 0.993), against 86.442% accuracy for Logistic Regression. A top-10 subset excluding BitcoinAddresses showed no statistically resolved accuracy difference from the full model, and a direct ablation without the feature changed accuracy by 0.009 points; smaller subsets lost accuracy. Because provenance, label semantics and external validity are unverified, the findings are a within-dataset baseline. On an external ransomware dataset (Table IV), the top-10 model reached 68.29% accuracy, so the findings do not show a deployable malware detector.

DATA AND CODE AVAILABILITY

The experiment script (Python, scikit-learn) reproduces the results in Tables I and II from the CSV using seeds 42, 43 and 44. Repository: https://github.com/naikam85/ransomware-pe-header-baseline. No binaries are distributed. The CSV is distributed on Kaggle under the Apache 2.0 licence, as stated on the dataset page. The external dataset [17] is distributed under CC BY 4.0, and the external test script is in the same repository.

ACKNOWLEDGEMENT

The authors thank their mentor, Prof. Poonam Surve, for her guidance during this work.

REFERENCES

[1] J. Saxe and H. Sanders, Malware Data Science: Attack Detection and Attribution. San Francisco, CA, USA: No Starch Press, 2018.

[2] T. Rezaei and A. Hamze, “An efficient approach for malware detection using PE header specifications,” in Proc. 6th Int. Conf. Web Research (ICWR), 2020, pp. 234–239, doi: 10.1109/ICWR49608.2020.9122312.

[3] A. Kharraz, W. Robertson, D. Balzarotti, L. Bilge, and E. Kirda, “Cutting the Gordian knot: A look under the hood of ransomware attacks,” in Proc. DIMVA, LNCS vol. 9148, 2015, pp. 3–24.

[4] E. Berrueta, D. Morato, E. Magaña, and M. Izal, “A survey on detection techniques for cryptographic ransomware,” IEEE Access, vol. 7, pp. 144925–144944, 2019.

[5] D. Arp et al., “Dos and don’ts of machine learning in computer security,” in Proc. 31st USENIX Security Symp., 2022, pp. 3971–3988.

[6] “MLRD: Machine Learning Ransomware Detection,” GitHub repository, securycore/MLRD-Machine-Learning-Ransomware-Detection. [Online]. Available: https://github.com/securycore/MLRD-Machine-Learning-Ransomware-Detection (commit b68fa301fad8385cff2c98547bc2b023a8b03158; accessed Oct. 27, 2026).

[7] H. S. Anderson and P. Roth, “EMBER: An open dataset for training static PE malware machine learning models,” arXiv:1804.04637, 2018.

[8] F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder, and L. Cavallaro, “TESSERACT: Eliminating experimental bias in malware classification across space and time,” in Proc. 28th USENIX Security Symp., 2019, pp. 729–746.

[9] C. Ambroise and G. J. McLachlan, “Selection bias in gene extraction on the basis of microarray gene-expression data,” Proc. Natl. Acad. Sci. USA, vol. 99, no. 10, pp. 6562–6566, 2002.

[10] amdj3dax, “Ransomware detection dataset,” Kaggle dataset. [Online]. Available: https://www.kaggle.com/datasets/amdj3dax/ransomware-detection-data-set (accessed Oct. 31, 2026).

[11] L. Breiman, “Random forests,” Mach. Learn., vol. 45, no. 1, pp. 5–32, 2001.

[12] F. Pedregosa et al., “Scikit-learn: Machine learning in Python,” J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011.

[13] C. Strobl, A.-L. Boulesteix, A. Zeileis, and T. Hothorn, “Bias in random forest variable importance measures: Illustrations, sources and a solution,” BMC Bioinformatics, vol. 8, art. 25, 2007.

[14] T. Saito and M. Rehmsmeier, “The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets,” PLoS ONE, vol. 10, no. 3, art. e0118432, 2015.

[15] D. Chicco and G. Jurman, “The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation,” BMC Genomics, vol. 21, art. 6, 2020.

[16] C. Nadeau and Y. Bengio, "Inference for the generalization error," Mach. Learn., vol. 52, no. 3, pp. 239–281, 2003.

[17] C. Moreira, D. Moreira, and C. Sales Jr., "Ransomware PE Header Feature Dataset," Mendeley Data, v3, 2024, doi: 10.17632/p3v94dft2y.3. [Online]. Available: https://data.mendeley.com/datasets/p3v94dft2y (accessed Oct. 31, 2026).

[18] "Improving Ransomware Detection based on Portable Executable Header using Xception Convolutional Neural Network," Comput. Secur. art. no. 103265, 2023, doi: 10.1016/j.cose.2023.103265.