International Journal of Recent Research and Review, Vol. XIX, Issue 3, September 2026
ISSN 2277 – 8322

COMPARATIVE EVALUATION OF CLASSICAL AND ENSEMBLE MACHINE-LEARNING CLASSIFIERS FOR ANDROID MALWARE CATEGORY CLASSIFICATION USING DYNAMIC BEHAVIOUR FEATURES

Amit Kumar Sanwariya[1], Kriti Sharma[2], Dr. Pradeep Jha[3]
[1,2,3]Department of Computer Science and Engineering
[1,2,3]Global Institute of Technology, Jaipur, Rajasthan, India

Abstract- Android malware is produced faster than it can be analysed manually, and analysts need to know not only whether an application is malicious but which category of malware it belongs to. This paper compares seven classical and ensemble classifiers for five-class categorisation (Adware, Banking, SMS malware, Riskware, Benign) using the system-call, binder-call and composite-behaviour frequencies of the CICMalDroid 2020 dataset (11,598 applications, 470 features). A random-forest importance selector reduced the feature space to 235 features, fitted on the training partition only, and all classifiers were evaluated on a stratified 20% hold-out set of 2,320 applications with accuracy, macro precision, recall and F1-score, specificity, Matthews correlation coefficient (MCC), Cohen's kappa, ROC-AUC and confusion matrices. Random Forest achieved the best results (94.83% accuracy, macro F1-score 0.938, MCC 0.933, ROC-AUC 0.995), followed by Extra Trees (94.48%) and Gradient Boosting (93.02%), whereas Naive Bayes reached only 60.56%. SMS malware was recognised almost perfectly, while most errors involved Adware, Riskware and Benign applications. File-system, memory-management and telephony-identifier calls were among the most informative features. The study provides a reproducible benchmark and shows that a fixed-parameter tree ensemble is a strong, inexpensive baseline for behaviour-based Android malware categorisation.

Keywords: Android malware; malware categorisation; CICMalDroid 2020; system calls; random forest; ensemble learning; feature selection

I. INTRODUCTION

Android is the most widely used mobile operating system and, because of its open distribution model, the principal target of mobile malware [1, 2, 3]. The volume and

diversity of malicious applications make manual inspection impractical, so automatic classification based on machine learning is widely used [4]. Deciding only whether an application is malicious is often not enough: adware, banking trojans, SMS malware and riskware differ in the assets they attack and in the response they require. Assigning an application to a malware category therefore supports triage, prioritisation and reporting.

Dynamic features, which describe what an application actually does at run time, are harder for malware authors to conceal than static features such as permissions or API calls [5, 6]. The CICMalDroid 2020 dataset [7] provides such features (frequencies of system calls, binder calls and composite behaviours) for thousands of labelled applications. Published results on this dataset use different feature subsets, protocols and metrics, which makes it hard to say which classifier family is preferable. The contributions of this paper are:

- a uniform, leakage-free pipeline (stratified split, feature selection and scaling fitted on the training data only) applied to seven classical and ensemble classifiers on the five-class CICMalDroid 2020 feature-vector file;
- a multi-metric evaluation (accuracy, macro precision/recall/F1, specificity, MCC, Cohen's kappa, ROC-AUC, confusion matrices) that goes beyond accuracy;
- an analysis of the most informative behavioural features and of the most frequent confusions between categories.

II. RELATED WORK

Surveys of machine-learning based Android malware detection cover static, dynamic and hybrid approaches and stress dataset imbalance, feature redundancy and adversarial evasion [1, 2, 3]. Dynamic analysis on real devices has been shown to be more reliable than emulation for detection [6, 8], and system-call and memory-access traces are discriminative features [5]. For CICMalDroid 2020, Amenova et al. [9] report 94% accuracy with a CNN-LSTM, and the dataset creators describe a semi-supervised deep-learning framework [7] and a later hybrid stacked auto-encoder approach [10].

Several comparative studies report that ensemble methods such as Random Forest and gradient boosting are the most consistent classifiers for structured malware features, with deep networks giving at best marginal gains at much higher cost [11, 12, 13]. This agrees with the wider observation that tree-based models remain very competitive on tabular data [14, 15]. Ensemble learning with

bagging, boosting and randomisation is described in [16, 17, 18, 19]. What is missing is a controlled comparison of these classifiers on the five-class CICMalDroid 2020 feature-vector file with several complementary metrics, which this paper provides.

III. MATERIALS AND METHODS

3.1 Dataset

The experiments use the feature-vector file of CICMalDroid 2020 [7], containing 11,598 applications described by 470 integer features that count the system calls, binder calls and composite behaviours observed during execution. All columns are integers and no values are missing. The five classes are Adware (1,253 applications), Banking malware (2,100), SMS malware (3,904), Riskware (2,546) and Benign (1,795); the largest class is therefore about three times the size of the smallest. Exploratory analysis showed heavily right-skewed features with extreme outliers, weak pairwise correlations with groups of moderately correlated features, and strong class overlap in every two-dimensional projection (Figure 1).

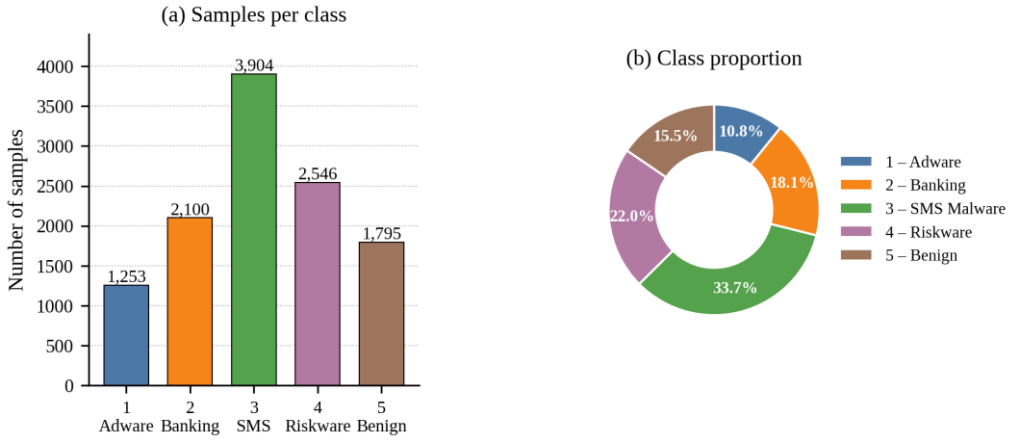


Figure 1. Class distribution of the dataset: (a) number of applications per class; (b) class proportions.

3.2 Pre-processing and feature selection

The data were split with a stratified hold-out (test fraction 0.2, random seed 42) into 9,278 training and 2,320 test applications (Table 1). A random forest of 100 trees was fitted on the training partition, and features with impurity-based importance at least equal to the median were retained (scikit-learn SelectFromModel, median threshold), reducing the feature space from 470 to 235. The selected features were standardised with a scaler fitted on the training partition. Selection and scaling

were fitted on training data only, so no information from the test partition entered the pipeline (Figure 2).

Table 1. Stratified partitions of the dataset.

Class	Category	Total	Training	Test
1	Adware	1,253	1,002	251
2	Banking	2,100	1,680	420
3	SMS malware	3,904	3,123	781
4	Riskware	2,546	2,037	509
5	Benign	1,795	1,436	359
	Total	11,598	9,278	2,320

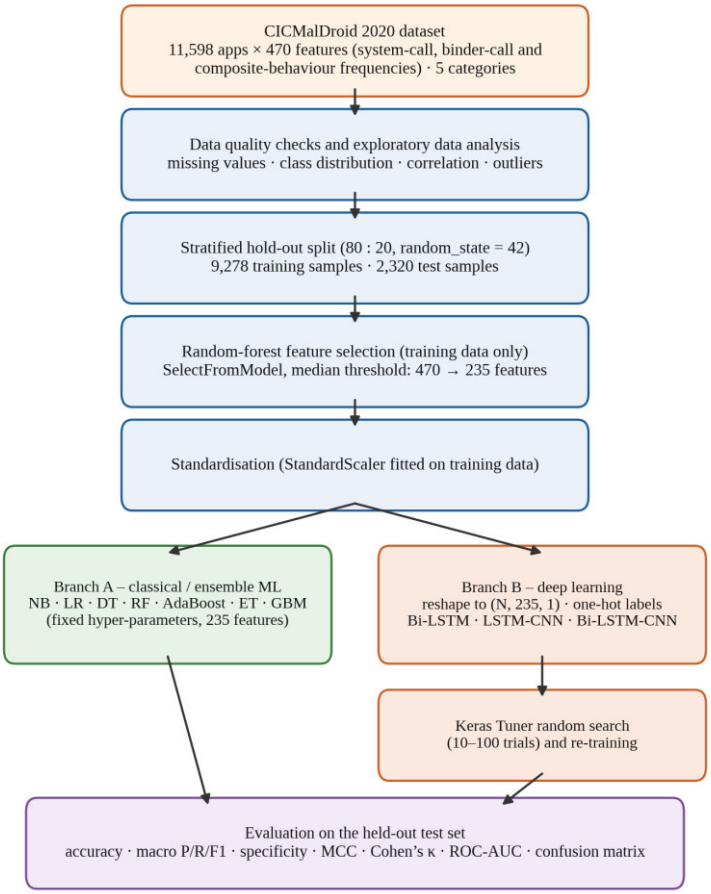


Figure 2. Experimental workflow (this paper uses the classical / ensemble branch).

3.3 Classifiers

Seven scikit-learn classifiers [20] were trained on the 235 standardised features: Gaussian Naive Bayes, Logistic Regression, a Decision Tree, Random Forest [16], AdaBoost [17], Extra Trees [18] and Gradient Boosting [19]. Hyper-parameters were fixed in advance and not tuned on the test set (Table 2).

Table 2. Classifier configuration (unlisted settings are scikit-learn defaults).

Classifier	Main settings
Naive Bayes	var_smoothing = 0.01
Logistic Regression	C = 100 (lbfgs; reported non-convergence within default iterations)
Decision Tree	max_depth = 15
Random Forest	200 trees, max_depth = 30, min_samples_split = 2, min_samples_leaf = 1
AdaBoost	base tree max_depth = 3, SAMME.R, learning_rate = 0.1
Extra Trees	defaults (100 trees)
Gradient Boosting	100 estimators, learning_rate = 0.1, max_depth = 3, min_samples_split = 10, min_samples_leaf = 5

3.4 Evaluation

All models were scored on the 2,320 test applications. Precision, recall, F1-score and specificity were computed per class from the confusion matrix and macro-averaged, so that each category counts equally. The Matthews correlation coefficient [21, 22] and Cohen's kappa [23] summarise the confusion matrix in a single chance-corrected value, and the ROC-AUC [24] was computed one-vs-rest and macro-averaged from predicted probabilities. Metric values other than ROC-AUC were recomputed from the confusion matrices.

IV. RESULTS

4.1 Feature importance

Figure 3 shows the 50 features with the highest random-forest importance. Importance decreases smoothly from about 0.022 to about 0.007, i.e. it is spread over many features. The highest-ranked features are file-system and memory-related system calls (pread64, access, stat64, lstat64, read, munmap, mkdir), time and signal calls

(gettimeofday, sigprocmask) and binder calls that query display, device, subscriber and package information (getDisplayInfo, getDeviceId, getSubscriberId, getLine1Number, getInstallerPackageName, getApplicationInfo), together with the composite behaviours ACCESS_PERSONAL_INFO and CREATE_FOLDER. The importance of telephony identifiers is plausible for categories that harvest device data, although importance indicates association and not causation.

ISSN 2277 – 8322

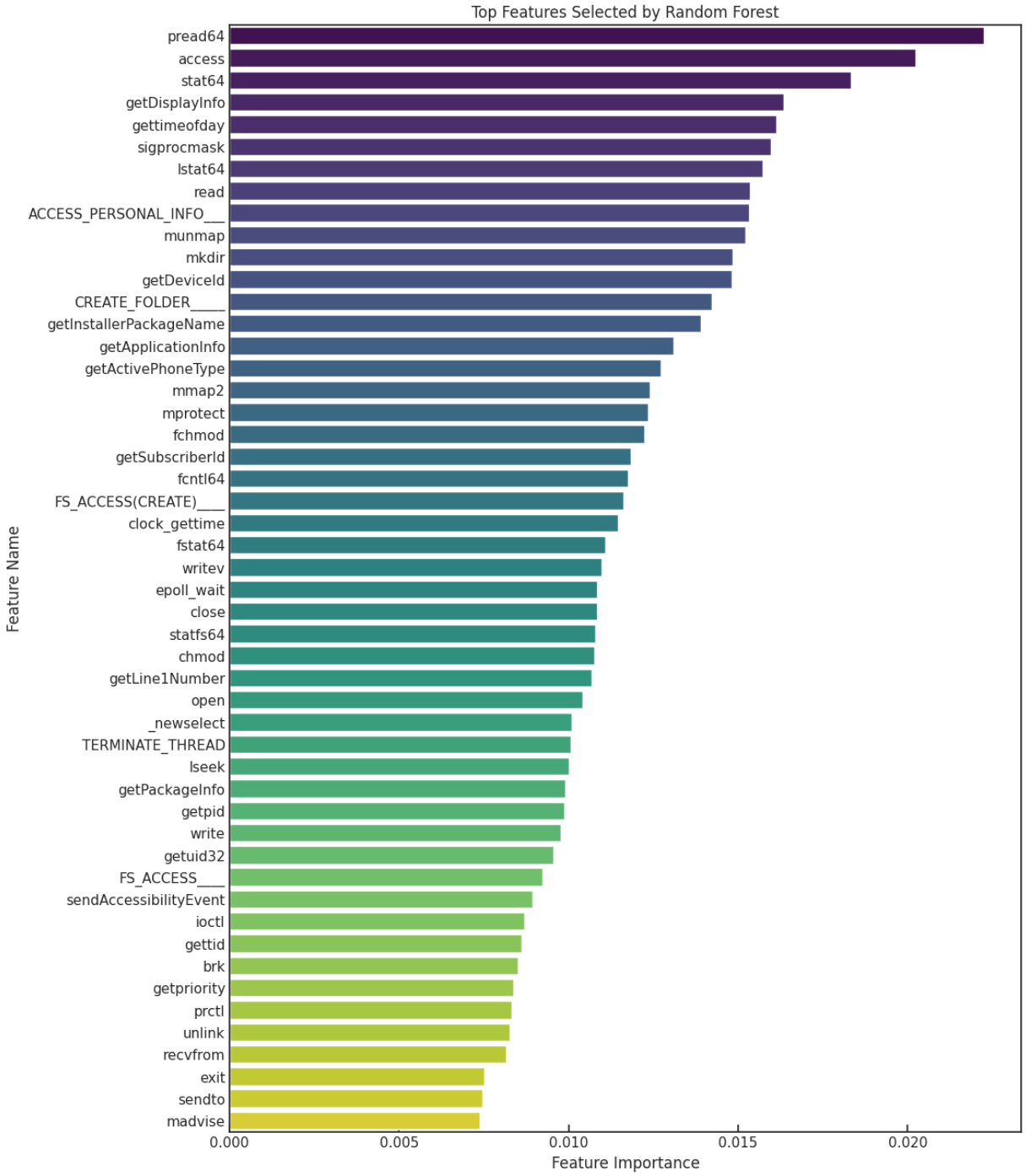


Figure 3. The 50 most important features according to the random forest.

4.2 Overall performance

Table 3. Test performance of the seven classifiers (2,320 applications).

Classifier	Train acc. (%)	Test acc. (%)	Macro P	Macro R	Macro F1	Macro spec.	MCC	κ	ROC-AUC
Naive Bayes	60.71	60.56	0.660	0.539	0.545	0.890	0.501	0.463	0.829
Logistic Regression	88.88	86.25	0.859	0.831	0.843	0.964	0.821	0.820	0.952
Decision Tree	98.74	89.78	0.877	0.878	0.877	0.975	0.867	0.867	0.913
Random Forest	99.85	94.83	0.937	0.941	0.938	0.987	0.933	0.933	0.995
AdaBoost	89.69	88.53	0.876	0.858	0.864	0.971	0.852	0.850	0.968
Extra Trees	99.85	94.48	0.933	0.938	0.935	0.986	0.929	0.928	0.994
Gradient Boosting	95.30	93.02	0.921	0.919	0.920	0.982	0.909	0.909	0.991

Table 3 and Figure 4 give the results. Random Forest was best on every summary metric (94.83% accuracy, macro F1-score 0.938, MCC 0.933, ROC-AUC 0.995). Extra Trees was within 0.35 percentage points and Gradient Boosting reached 93.02%. The single Decision Tree over-fitted (98.74% training against 89.78% test accuracy). AdaBoost (88.53%) and Logistic Regression (86.25%) had small train–test gaps but lower accuracy, and Naive Bayes (60.56%) performed poorly, as expected when strongly correlated count features violate its independence assumption.

ISSN 2277 – 8322

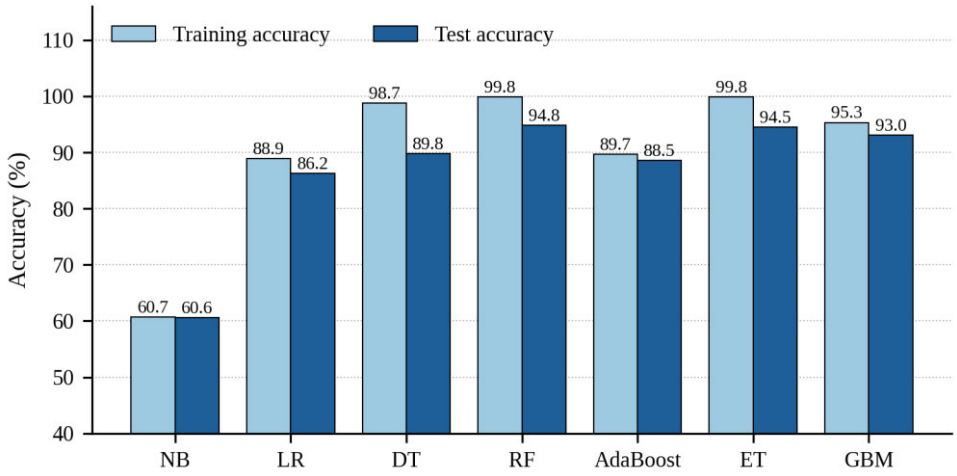


Figure 4. Training and test accuracy of the classifiers.

4.3 Per-class behaviour and errors

Table 4. Per-class results of the Random Forest.

Class	Category	Precision	Recall	F1-score	Support
1	Adware	0.877	0.936	0.906	251
2	Banking	0.969	0.900	0.933	420
3	SMS Malware	0.975	0.994	0.984	781
4	Riskware	0.965	0.919	0.942	509
5	Benign	0.900	0.955	0.927	359

SMS malware is recognised almost perfectly by every competent classifier (F1-score 0.91–0.98), whereas Adware, the smallest class, is generally the hardest (Figure 6, Figure 5). The most frequent Random Forest errors are Riskware predicted as Benign (20 applications, 3.9% of Riskware) or Adware (16; 3.1%) and Banking predicted as Adware (15; 3.6%). Naive Bayes assigns most applications to SMS malware (recall 98.5%, precision 55.2%). Aggregating the four malicious classes, the Random Forest labelled 38 of 1,961 malicious applications (1.9%) as Benign and 16 of 359 benign applications (4.5%) as malicious.

ISSN 2277 – 8322

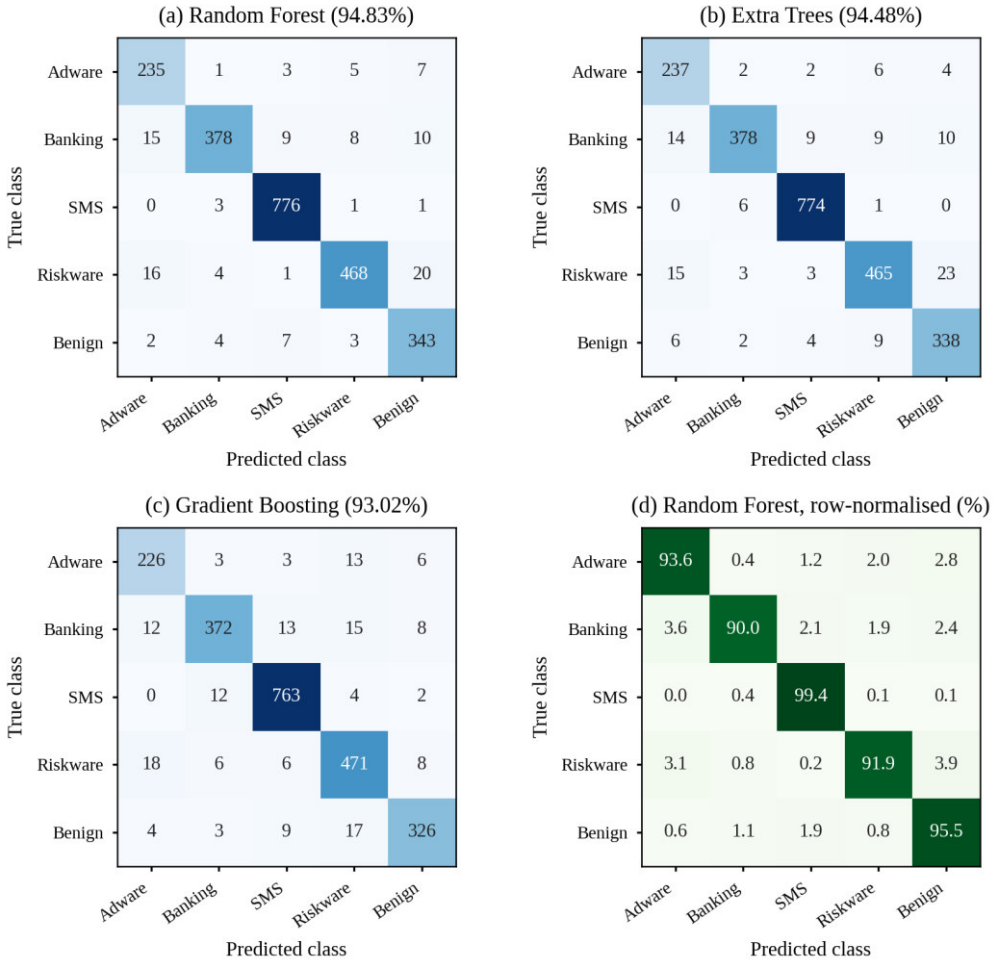


Figure 5. Confusion matrices of (a) Random Forest, (b) Extra Trees, (c) Gradient Boosting and (d) the row-normalised Random Forest matrix.

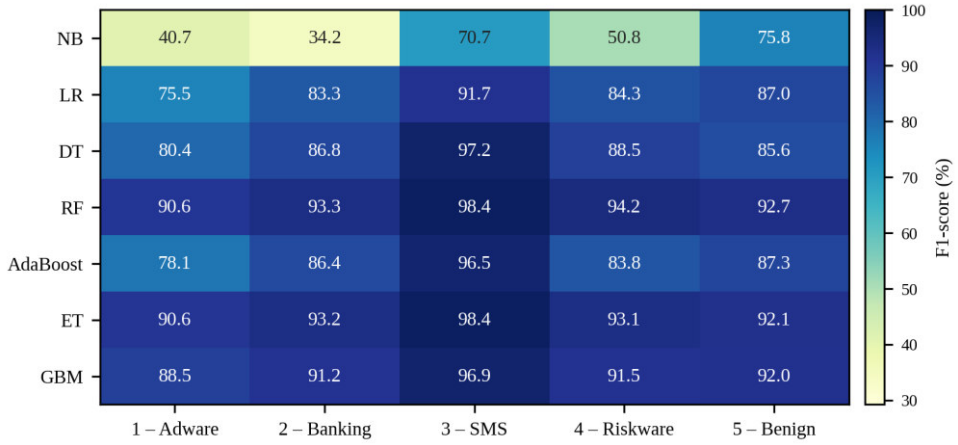


Figure 6. Per-class F1-score (%) of the seven classifiers.

V. DISCUSSION

The ordering Random Forest > Extra Trees > Gradient Boosting > Decision Tree > AdaBoost > Logistic Regression > Naive Bayes is the same for accuracy, macro F1-score, MCC and kappa, so the conclusion does not depend on the metric. Averaging many decorrelated trees lowers the variance of a single tree and copes with skewed counts, outliers and correlated features without scaling assumptions. The confusion among Adware, Riskware and Benign applications is plausible because these categories can behave much like ordinary applications, and it suggests that additional information, such as static features, would be needed to separate them further. Amenova et al. [9] report 94% for a CNN-LSTM on the same dataset; our 94.83% for a Random Forest is of the same order, although protocols differ and the values are not strictly comparable. From a practical viewpoint the ensemble needs no GPU or tuning search and provides feature importances directly.

VI. LIMITATIONS AND FUTURE WORK

Results come from a single stratified hold-out split with one random seed, so confidence intervals are not available; repeated or nested cross-validation would strengthen the comparison. The classifiers used fixed hyper-parameters that were not tuned, so their ranking may change with tuning. Feature importance is a global, impurity-based measure, and explanation of individual decisions (for example with SHAP [25]) is future work, as are adversarial-robustness testing [26], evaluation on newer malware and gradient-boosting libraries such as XGBoost and LightGBM.

VII. CONCLUSION

On the five-class CICMalDroid 2020 feature-vector task, a Random Forest trained on 235 selected features gave the best results (94.83% accuracy, MCC 0.933, ROC-AUC 0.995), with Extra Trees and Gradient Boosting close behind, while simple probabilistic and linear models lagged clearly. Errors concentrate on Adware, Riskware and Benign applications, and the most informative features are file-system, memory and telephony-identifier calls. A fixed-parameter tree ensemble is therefore a strong and inexpensive baseline for behaviour-based Android malware categorisation.

REFERENCES

[1] K. Liu, S. Xu, G. Xu, M. Zhang, D. Sun, and H. Liu, “A review of Android malware detection

approaches based on machine learning," *IEEE Access*, vol. 8, pp. 124579–124607, 2020.

[2] V. Kouliaridis and G. Kambourakis, "A comprehensive survey on machine learning techniques for Android malware detection," *Information*, vol. 12, no. 3, p. 118, 2021.

[3] A. Muzaffar, H. R. Hassen, M. A. Lones, and H. Zantout, "An in-depth review of machine learning based Android malware detection," *Computers & Security*, vol. 121, p. 102833, 2022.

[4] N. Z. Gorment, A. Selamat, L. K. Cheng, and O. Krejcar, "Machine learning algorithm for malware detection: Taxonomy, current challenges, and future directions," *IEEE Access*, vol. 11, pp. 107539–107552, 2023.

[5] Z. Xu, S. Ray, P. Subramanyan, and S. Malik, "Malware detection using machine learning based analysis of virtual memory access patterns," in *Proc. Design, Automation & Test in Europe Conf. & Exhibition (DATE)*, Lausanne, Switzerland: IEEE, 2017, pp. 169–174.

[6] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "DL-Droid: Deep learning based Android malware detection using real devices," *Computers & Security*, vol. 89, p. 101663, 2020.

[7] S. Mahdavifar, A. F. A. Kadir, R. Fatemi, D. Alhadidi, and A. A. Ghorbani, "Dynamic Android malware category classification using semi-supervised deep learning," in *Proc. IEEE Int. Conf. on Dependable, Autonomic and Secure Computing (DASC/PiCom/CBDCom/CyberSciTech)*, 2020, pp. 515–522.

[8] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "Emulator vs real phone: Android malware detection using machine learning," in *Proc. 3rd ACM Int. Workshop on Security and Privacy Analytics (SPA)*, New York, NY, USA, 2017, pp. 65–72.

[9] S. Amenova, C. Turan, and D. Zharkynbek, "Android malware classification by CNN-LSTM," in *Proc. Int. Conf. on Smart Information Systems and Technologies (SIST)*, 2022, pp. 1–4.

[10] S. Mahdavifar, D. Alhadidi, and A. A. Ghorbani, "Effective and efficient hybrid Android malware classification using pseudo-label stacked auto-encoder," *Journal of*

Network and Systems Management, vol. 30, art. 22, 2022.

[11] M. S. Akhtar and T. Feng, "Malware analysis and detection using machine learning algorithms," *Symmetry*, vol. 14, no. 11, p. 2304, 2022.

[12] M. S. Akhtar and T. Feng, "Evaluation of machine learning algorithms for malware detection," *Sensors*, vol. 23, no. 2, p. 946, 2023.

[13] M. Azeem, D. Khan, S. Iftikhar, S. Bawazeer, and M. Alzahrani, "Analyzing and comparing the effectiveness of malware detection: A study of machine learning approaches," *Heliyon*, vol. 10, no. 1, p. e23365, 2024.

[14] L. Grinsztajn, E. Oyallon, and G. Varoquaux, "Why do tree-based models still outperform deep learning on typical tabular data?" in *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2022.

[15] R. Shwartz-Ziv and A. Armon, "Tabular data: Deep learning is not all you need," *Information Fusion*, vol. 81, pp. 84–90, 2022.

[16] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[17] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[18] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 63, no. 1, pp. 3–42, 2006.

[19] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

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

[21] B. W. Matthews, "Comparison of the predicted and observed secondary structure of T4 phage lysozyme," *Biochimica et Biophysica Acta*, vol. 405, no. 2, pp. 442–451, 1975.

[22] J. Gorodkin, "Comparing two K-category assignments by a K-category correlation coefficient," *Computational Biology and Chemistry*, vol. 28, no. 5–6, pp. 367–374, 2004.

[23] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and*

Psychological Measurement, vol. 20, no. 1, pp. 37–46, 1960.

[24] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

[25] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 4765–4774.

[26] L. Demetrio, S. E. Coull, B. Biggio, G. Lagorio, A. Armando, and B. Nelson, "Adversarial EXEmples: A survey and experimental evaluation of practical attacks on machine learning for Windows malware detection," *ACM Transactions on Privacy and Security*, vol. 24, no. 4, pp. 1–34, 2021.