

Android Malware Detection: Feature Update Using Incremental Learning Approach: Further Investigation of UFILA

ZAKARIA SAWADOGO ^{*†}, JEAN-MARIE DEMBELE[†], GERVAIS MENDY ^{*}, SAMUEL OUYA ^{*}.

^{*}LITA(Laboratory of Computer Science, Telecommunications and Applications), Cheikh Anta Diop University of Dakar, Senegal

[†]LANI(Laboratory of Numerical Analysis and Computer Science), Gaston Berger University, Senegal

{sawadogo.zakaria,jean-marie.dembele}@ugb.edu.sn, {gervais.mendy,samuel.ouya}@ucad.edu.sn

Abstract—Several kinds of mobile applications are available on platforms offering various services to users. Both malware and good applications are found in software repositories, which is a major cybersecurity problem. To address this problem, machine learning approaches have been proposed in the literature for the detection of malware in general and malicious Android applications in particular. But obfuscation techniques are used by some developers to hide malicious applications, which implies the need to update Android malware detection models. In the literature few works are focused on updating features. Our contribution is therefore an incremental learning approach capable of detecting Android malware by taking into account both static and dynamic features. We propose through the UFILA approach an updated feature set for Android malware detection and classification by adding new features. We evaluated 13 classification algorithms and selected the four most effective algorithms to implement our approach. The results obtained by our approach outperform several malware detection approaches in the literature. Our approach allows to detect malicious android applications based on static and dynamic features with feature updating. It also allows to efficiently classify the different classes.

Index Terms—imbalanced dataset, Android malware detection, Malware classification, Artificial intelligence, Machine learning.

I. INTRODUCTION

Over the past decade, we have seen an exponential growth in the use of smartphones. The report of Digital 2021 of "We are Social"[1], already indicated that in January 2022, there would 5.31 billion mobile users in the world. This number represented 67.1% of the total world population and this corresponds to an increase of 1.8% compared to January 2020. This growth goes hand in hand with those of mobile applications. The mobile system Android exploitation in particular which represents 80.8% of mobile operating system users according to the Digital 2021 report by We are Social[1]. Growth very often leads to setbacks. We find that the growth in the use of mobile applications in our daily tasks has also allowed

the development of malicious Android mobile applications. According to Statista 2021[2], from June 2016 to March 2020, the total number of new Android malware was 482,579 per month. In addition, according to the same report, over than 25% of mobile users were attacked by malware in the third quarter of 2020. Algeria, Nigeria, Morocco were the most affected African countries with respectively 16.28%, 13.31% and 11.12% of recorded attacks.

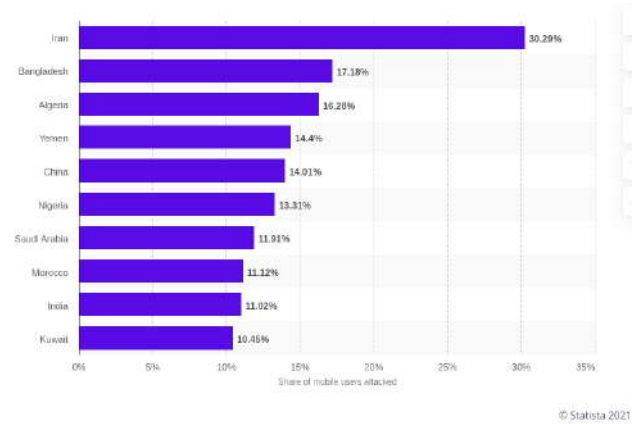


Fig. 1. Countries with highest mobile malware encounter rate Q3 2020.[2]

This growth in cybercrime has triggered a lot of research to detect or classify malicious Android apps. The searches can be classified into two categories namely those using artificial intelligence algorithms including machine learning and deep learning and those using other algorithms such as the similarity algorithms(Scoring Algorithm) or Signature based [3] [4] or Integrated Detection Mechanisms[5]. Several analysis techniques are used by researchers including static, dynamic and water analysis for the classification or detection of malicious Android applications (Sihag, Vardhan, and Singh 2021) [6]. The introduction of artificial intelligence methods, such as machine learning, has improved the outlook for Android malware classification and detection. But According to Zou et al. 2020 [7], Vasan et al. 2020[8] and Rathor et al.2020 [9], several malicious application developers use obfuscation techniques such as insertion of dead code, reorganization of subroutines, decryption transposition and wrapping to hide malware or create variants of a existing malware family in order to evade detection. This makes

Manuscript received September 15, 2022.

This work was supported by the Partnership for Applied Science, Engineering and Technology Skills - Regional Scholarship and Innovation Fund (PASET-RSIF), and is a follow-up to the invited review of the accepted and presented paper entitled "Android Malware Classification: Updating Features Through Incremental Learning Approach (UFILA)" of the 24th International Conference on Advanced Communication Technologies (ICACT2022).

Zakaria SAWADOGO, researcher in cybersecurity including artificial at Gaston Berger University,(corresponding author, email: sawadogo.zakaria@ugb.edu.sn)

Pr. Jean-Marie DEMBELE is Assimilated Professor in Computer Science at Gaston Berger University, (email: jean-marie.dembele@ugb.edu.sn)

Pr. Gervais MENDY is a researcher-scientist at the ESP polytechnic school at Cheikh Anta Diop University , (email: gervais.mendy@ucad.edu.sn)

Pr. Samuel OUYA is currently the director of the LITA laboratory at Cheikh Anta Diop University , (email: samuel.ouya@ucad.edu.sn)

models obsolete because they cannot detect vulnerabilities that are not taken into account in the training of the model [6] [10]. It will be necessary to add new features during the training process that allow the updating of the models. It is in this sense that the authors (Sanghani and Kotecha 2019) [11] developed an incremental personalized email spam filter, which is an incremental learning framework using Support Vector Machine (SVM) so as to update the features. In this approach the features are selected before the incremental learning process. So the possible impact of the non selected features between a precedent step and a next step is not considered. Then we can not make correlation between the initial set of features and the new features. Published research on this issue of updating features is very rare. This justifies our contribution. We propose an extension Updating Features Through Incremental Learning Approach (UFILA) [12], which allows the addition of features. Thus, this work focuses on an incremental learning approach that allows updating of features for the classification of malicious Android applications. In this proposal, we are the following contributions:

- 1) The proposal of UFILA, which is an efficient and functional system for classifying and identifying Android malware, enabling the features updating.
- 2) The evaluation of the effectiveness of UFILA using classification algorithms: Extra Trees Classifier (et), Random Forest (rf) Classifier, Light Gradient Boosting Machine (lightgbm), Decision Tree Classifier (dt).
- 3) Our approach allows to detect malicious android applications based on static and dynamic features with feature updating. It also allows to efficiently classify the different classes.

The paper is structured as follows: After this introduction, in section II we discussed the state of the art of Android malware classification and incremental learning. In section III, we detailed our approach and methodology. Section IV allowed us to present the different elements of our experimental process: the dataset, the experience environment and the metrics used to measure the performance of our approach. In section V, we presented and analyzed the experiential results. We end the paper by a conclusion in the last section VI.

II. STATE OF THE ART

In this section we discuss through previous work on the approaches of classifications and incremental learning.

A. Android malware classification using machine learning

Classification is a task that requires the use of machine learning algorithms to assign a class label based examples in the problem domain. In the literature several researchers have used classification algorithms to classify malicious android applications. (D. Li et al. 2017) [13] presented a fine-grained Android malware detection mechanism. They used Drebin datasets with 5560 malware from different families. Various classification algorithms such as Gaussian Bayes, Bernoulli Bayes, Decision Tree, Random forest, Logistic Regression and

SVM have been tested. Random Forest showed the greatest accuracy which is 98.61%.

(Wei et al. 2017) [14] proposed a real-time malware detection tool called AndroidDetect. The tool is based on the analysis of the relationship between system functions, permissions and API calls. It applies many machine learning classifiers, such as decision tree, Naive Bayesian and J48. (Liu et al. 2018) [15] proposed an Android malware detection model based on LightGBM. The model consists of a new feature selection method, which contains Chi2 and Extra Trees, and a LightGBM classification method. A corpus of 2000 malware and an equal number of benign samples are prepared to verify the model. The results show a high accuracy of the model which is 96.4%

(Arora et al. 2018) Arora2018 have proposed a hybrid malware detection system by combining a supervised and unsupervised method. In the detection engine, the supervised method (KNN) and the unsupervised method (K-medoids) are combined and provided an accuracy rate of 91.98 %. However, an improvement in the level of precision is still necessary. In addition, the number of samples in the training and test data set is relatively small.

(Kedziora M et al. 2019) [16] have developed a detection model using five classifiers and three selection algorithms. A number of 1958 requests were reviewed, and the 996 are malware with an accuracy level of 80.7%.

(Modallal 2020) [17] proposed MLSecAndroid which is a tool capable of identifying Android malware using a combination of machine learning classification techniques and the number of features extracted (permissions, calls API, user comments, etc). He created a dataset that contains 17,100 Android applications (14,984 benign applications and 2,116 software applications). And achieved 98.21% accuracy and 93.52% recall

(Alazab et al. 2020) [18] proposed an efficient classification model that combines authorization requests and API calls. Android applications using a large number of APIs. The model has three different grouping strategies to choose the most valuable API calls to maximize the likelihood of identifying malicious Android apps. The empirical results obtained with a true malware dataset containing 27,891 Android applications is F1-score of 94.3%.

B. Learning incremental

Incremental learning is a method of machine learning in which the input data is continuously used to extend the knowledge of the existing model, i.e. to further train the model [19] [20]. It represents a dynamic technique of supervised learning and unsupervised learning that can be applied when training data becomes progressively available over time or its size is out of the limits of system memory. (Brooks and Manza 2013) [21] added that, incremental learning is a machine learning paradigm in which the learning process takes place whenever new examples emerge and adjusts what has been learned based on new examples. (Marc Masana et al, 2020) [22] asserted in their survey that the future of machine learning

systems is incremental learning. Machine learning algorithms can be used for incremental learning according to (unwei ZHONG et al, 2017)[23].

(Chefrour 2019) [24] approaches in the same direction by saying that the well known methods as effective in the context of static machine learning do not offer an alternative for evolution and dynamic adaptation to integrate new data or to restructure already partially learned problems. In this field, incremental learning represents an interesting alternative and constitutes an open field of research. Algorithms that can facilitate incremental learning are known as incremental machine learning algorithms [23][24]. We have several works using classification algorithms to implement incremental learning. Here we present some of those works.

(Li et al. 2020)[25] have developed an incremental learning method based on the Multiclass SVM called "IMCSVM", which gradually improves classification ability by learning new malwares samples. Their incremental learning method is to add new classes to the existing classification model.

(Lopez-Lopez, Regueiro, and Pardo 2020) [26] proposed a new incremental learning approach that combines a deep functionality encoder with an open dynamic set of SVM, to solve the problem of identification people of interest (IoI) from streaming face data. (Sanghani and Kotecha 2019) [27] developed an incremental custom email spam filter, that is an incremental learning framework using SVM which overcomes the limitation of a static decision model to handle drift concept and deeply improves the performance of the classifier.

It emerges from the state of the art on incremental learning that most of the proposed approaches only update the data. The proposed approaches allow either to solve a lack of data, or for a problem of large amount of data that cannot be supplied to the model all of a sudden. So it's a horizontal update that means an update of the data. The explored literature did not address vertical updates that involve features. In UFILA approach, our proposal is for a vertical update.

III. UFILA APPROACH

A. Architecture and Methodology

In this section we first presented our approach which allows updating of features through an incremental learning approach called UFILA presented in figure 2. UFILA is an approach for classifying malicious and benign Android applications. UFILA is a vertical approach, so it naturally allows the updating of the features. Therefore the malware classification model is also updated. For the implementation of our approach, we have evaluated thirteen classification algorithms and selected three that we found to perform better. The different components of our architecture are: Features ,Features Update and Backup function ,Data Pre-processing, Training and Model, Testing and Prediction result.

Features, The data set D, made up of four families of characteristics, is subdivided into four batches noted d_1, d_2, d_3, d_4 .

Features Update and Backup function, The update and save function allows first to recover one by one the copies of the d_i by concatenating them progressively. The updating

of the features and the saving of the copy of the concatenation are carried out. **Data Pre-processing**, The pre-processing stage cleans up the data, scales it and subdivides it into two groups: the training set and the test set. **Training and Model**, This step allows to train algorithms in order to obtain a classification model. **Testing and Predictions result** ; In this step we test the performance of the obtained model with other data different from those used for training.

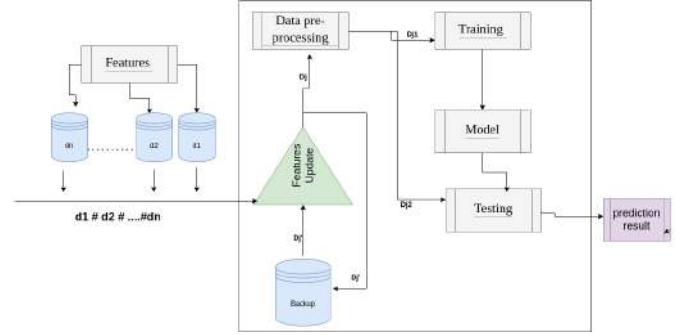


Fig. 2. Architecture of the UFILA approach

The different steps of our methodology are the following:

- 1) At initialization, the data batch d_1 arrives at the **Features Update** level, and we save d_1 in D_1 ($D_1 = d_1$). We make a copy of D_1 in D'_1 and D_1 goes to the **Data pre-processing** stage.
- 2) We perform the pre-processing at the **Data pre-processing** level, the dataset D_1 is divided into two parts D_{11} and D_{12} , where D_{11} is the Train Set and D_{12} is the Test Set.
- 3) The Train Set is used in the **Training** step to train the model.
- 4) At the **Testing** step, the Test Set is used to test the Model, resulting from the Training step.
- 5) The **Prediction result** step gives the performance of the model.
- 6) We will iterate the process described above. For that d_j with $j \geq 2$, is concatenated with D'_{j-1} to give D_j . A copy D'_j of D_j is made at the level of the Backup component. With D_j obtained, the steps 2,3,4,5 will be traversed. We stop the process when the iterations with D_n are done.

B. Function features update

In this subsection, we define and describe the concatenation function which is treated in the Features Update component in figure 2.

d_i : is a set of features, $1 \leq i \leq n$.

F_{ik} : is the K the feature of the data set d_i . $1 \leq k \leq m_i$. The features F_{ik} are pairwise different.

$d_1 = F_{11}, F_{12}, \dots, F_{1m_1}$

$d_2 = F_{21}, F_{22}, \dots, F_{2m_2}$

.

.

$$\begin{aligned}
& \cdot \\
d_j &= F_{j1}, F_{j2}, \dots, F_{jn_j} \\
& \cdot \\
& \cdot \\
& \cdot \\
d_n &= F_{n1}, F_{n2}, \dots, F_{nm_n}
\end{aligned}$$

By construction we have, we therefore have :

$$\begin{aligned}
D_1 &= d_1 = F_{11}, F_{12}, \dots, F_{1n_1} \\
D_2 &= d_2 \text{ Concat } D_1 \\
&= (F_{11}, F_{12}, \dots, F_{1n_1}) \text{Concat } (F_{21}, F_{22}, \dots, F_{2n_2}) \\
& \cdot \\
& \cdot \\
& \cdot \\
D_j &= d_j \text{ Concat } d_{j-1} \\
& \cdot \\
& \cdot \\
& \cdot \\
D_n &= d_n \text{ Concat } d_{n-1}
\end{aligned}$$

C. Selection of the classification algorithm

The classification algorithms are diverse depending on the desired goal. The authors [28] performed a study to determine the performance of supervised and unsupervised classification algorithms without giving the ranking. For the implementation of our approach, we have selected through the literature thirteen classification algorithms that we tested in order to select the best for our model. These are: Random Forest Classifier, SVM-Linear Kernel, Quadratic Discriminant Analysis, Naive Bayes, Ridge Classifier, Linear Discriminant Analysis, Logistic Regression, Ada Boost Classifier, Ada Boost Classifier, Decision Tree Classifier, Light Gradient Boosting Machine, Extra Trees Classifier, Gradient Boosting Classifier. Figure 8 shows the results of some metrics obtained after training on our dataset.

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
et	Extra Trees Classifier	0.9625	0.9861	0.9266	0.9704	0.9480	0.9187	0.9193	0.2440
rf	Random Forest Classifier	0.9619	0.9884	0.9251	0.9704	0.9471	0.9174	0.9181	0.1890
lightgbm	Light Gradient Boosting Machine	0.9565	0.9878	0.9130	0.9674	0.9393	0.9054	0.9065	0.1420
dt	Decision Tree Classifier	0.9507	0.9632	0.9202	0.9449	0.9323	0.8935	0.8938	0.0360
knn	K Neighbors Classifier	0.9487	0.9756	0.9210	0.9389	0.9298	0.8894	0.8895	0.4570
gbc	Gradient Boosting Classifier	0.9324	0.9760	0.8602	0.9518	0.9036	0.8517	0.8544	0.4070
lr	Logistic Regression	0.9321	0.9766	0.8808	0.9315	0.9053	0.8524	0.8534	0.2500
svm	SVM - Linear Kernel	0.9248	0.0000	0.8862	0.9131	0.8975	0.8383	0.8409	0.0660
ada	Ada Boost Classifier	0.9202	0.9725	0.8615	0.9175	0.8883	0.8264	0.8276	0.1210
lda	Linear Discriminant Analysis	0.9156	0.9672	0.8335	0.9307	0.8793	0.8148	0.8178	0.0700
ridge	Ridge Classifier	0.9147	0.0000	0.8301	0.9313	0.8777	0.8126	0.8158	0.0250
qda	Quadratic Discriminant Analysis	0.5891	0.6740	0.9943	0.4733	0.6413	0.2820	0.4007	0.0790
nb	Naive Bayes	0.5808	0.6683	0.9907	0.4682	0.6358	0.2695	0.3872	0.0460

Fig. 3. Selection of the classification algorithm

As one can see Light Gradient Boosting Machine Random Forest, Extra Trees classifier and Decision Tree Classifier, classifiers perform better in accuracy, precision, and F1-score. Extra Trees Classifier has the highest accuracy value at 96.25% and the other classifiers around 95%, so we choose

these four above mentioned classifiers for the implementation of our incremental approach.

IV. EXPERIMENTAL PROCESS

In this part we first present the dataset we used, the working environment, the tools and methods used to evaluate our approach.

A. Datasets

1) *Dataset used in binary classification* : The dataset is made up of vectors of 214 features extracted from 15,036 applications (5,560 malicious applications from Drebin Project[29] and 9,476 benign applications). The dataset had previously been used to develop and evaluate a multilevel classifier fusion approach for malware detection[30]. For the needs of our experience, we have to separate the features by category, and we obtain four datasets. d_1 : it contains 6 features of Intent, d_2 : it contains 23 features of API call signature, d_3 : it contains 72 features of Commands signature, d_4 : it contains 113 features Manifest Permission.

2) *Dataset used in multi-label classification*: We used the CCCS-CIC-AndMal-2020[31], dataset which includes 200,000 benign and 200,000 malware samples, for a total of 400,000 Android applications, with 12 malware categories including adware, backdoor, file infector, Potentially Unwanted Apps (PUA), ransomware, riskware, scareware, trojan, trojan-banker, trojan-dropper, trojan-sms and trojan-spy. Six feature categories were extracted after running the malware in an emulated environment. The main features extracted are as follows: Memory= d_1 : Memory characteristics define the activities performed by the malware using memory. API= d_2 : Application Programming Interface (API) characteristics delineate the communication between two applications. Network= d_3 : Network characteristics describe the data transmitted and received between other devices on the network. They indicate the network usage in the foreground and background. Battery= d_4 : Battery characteristics describe access to the battery wakelock and services by malware. Logcat= d_5 : Logcat features write log messages corresponding to a function performed by the malware. Processes= d_6 : The process features count the interaction of the malware with the total number of processes.

B. Experience environment

We conducted our experiment on an HP Z620 workstation with the following characteristics: Intel (R) Xeon (R) CPU E5-2643 0 @ 3.30 GHz 3.30GHz (2 processors), RAM: 34 GB, on which we installed Windows 10, 64 bit operating system. We used Anaconda Navigator which is a graphical user interface (GUI) which integrates several development tools such as Jupiter, Spyder and allows the use of python 3.6 and Scikit-learn, a free Python library for machine learning.

C. Classification Metrics

In machine learning and more specifically in classification problem, a confusion matrix, also called an error matrix, is a specific table layout that allows to visualize the performance of a supervised or unsupervised learning approach [32].

True Positive TP: Positive samples correctly classified by the model;

False Negative FN: A positive sample that is mis-classified by the model;

False Positive FP: A negative samples that is mis-classified by the model;

True Negative TN: Negative samples correctly classified by the model;

Additionally, the following metrics can be calculated from the confusion matrix:

Accuracy: Ratio of the number of correctly classified samples to the total number of samples for a given test data set. When classes are balanced, this is a good measure; if not, this metric is not very useful.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \quad (1)$$

It calculates the ratio of all “correctly detected items” to all “actually detected items”.

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

Recall or True Positive Rate TPR: It calculates the ratio of all “correctly detected items” to all “items that should be detected”.

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

F1-score: It calculates the harmonic mean of the precision and the recall.

$$F1 - score = 2 * \frac{Precision * Recall}{(Precision + Recall)} \quad (4)$$

These standard performance metrics from the confusion matrix were used for the evaluation of the model of our approach.

V. EXPERIMENTAL RESULTS AND ANALYSIS

A. Results and Analysis with binary classification

In this section we presented the results we obtained after implementation of our approach. The figure 20 shows the details obtained by our model with the different types of algorithms used for implementation

$D_1 = d_1$

$D_2 = d_2 \text{ Concat } D_1$

$D_3 = d_3 \text{ Concat } D_2 = d_1 \text{ Concat } d_2 \text{ Concat } d_3$

$D_4 = d_4 \text{ Concat } D_3 = d_1 \text{ Concat } d_2 \text{ Concat } d_3 \text{ Concat } d_4$

1) *Results and Analysis with binary classification: Classification report with D1* : Figures 4-7 shows the precision, recall, and F1-score metrics obtained by UFILA with D_1 . We can say that all metrics vary between 59% and 79% for any of the four use algorithms . We note that for all four algorithms, the values of the precision and F1-score metrics for the malware class are lower than those for the benign class. We have the opposite with the recall metric.

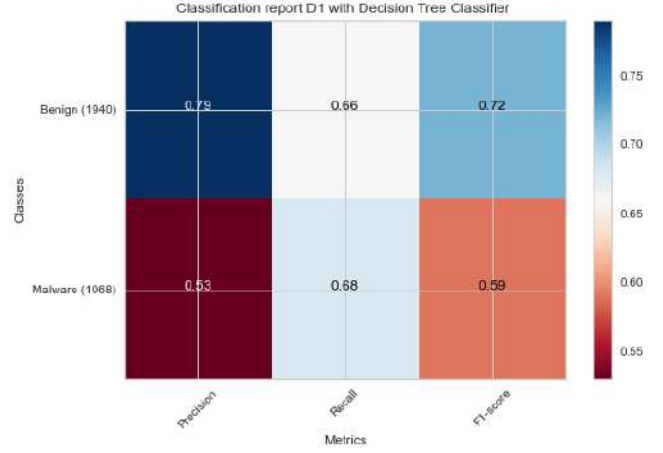


Fig. 4. Classification report D1 with Decision Tree Classifier

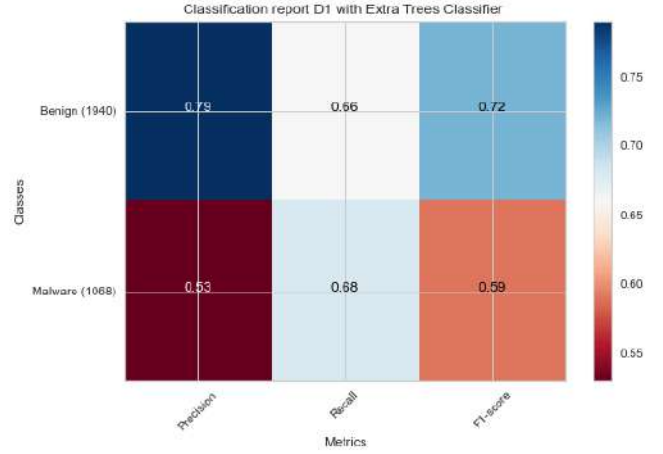


Fig. 5. Classification report D1 with Extra Trees Classifier

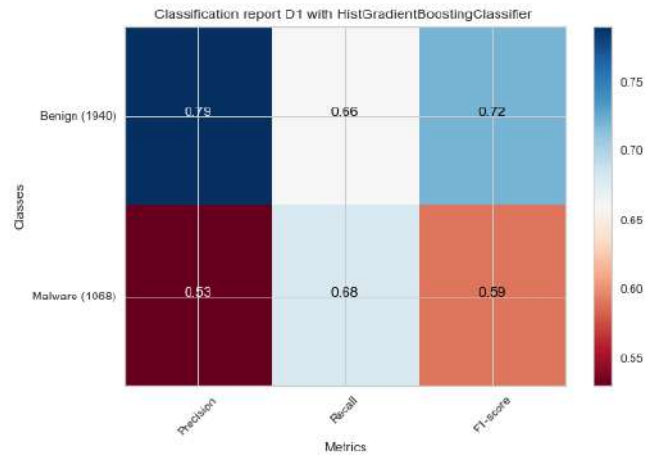


Fig. 6. Classification report D1 with Hist Gradient Boosting Classifier

2) *Results and Analysis with binary classification: Classification report with D2*: In figures 8 -11 , we obtained with

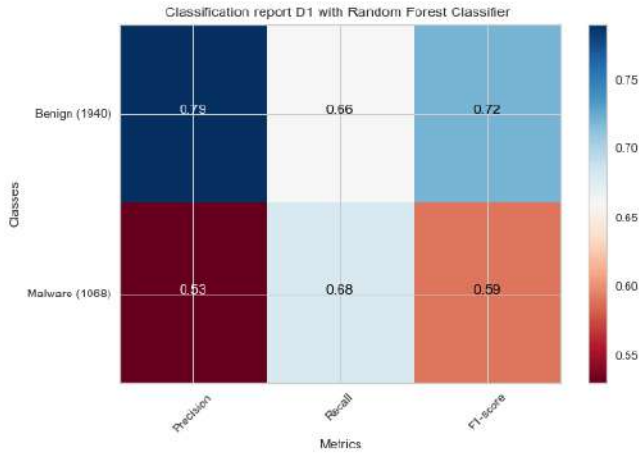


Fig. 7. Classification report D1 with Random Forest Classifier

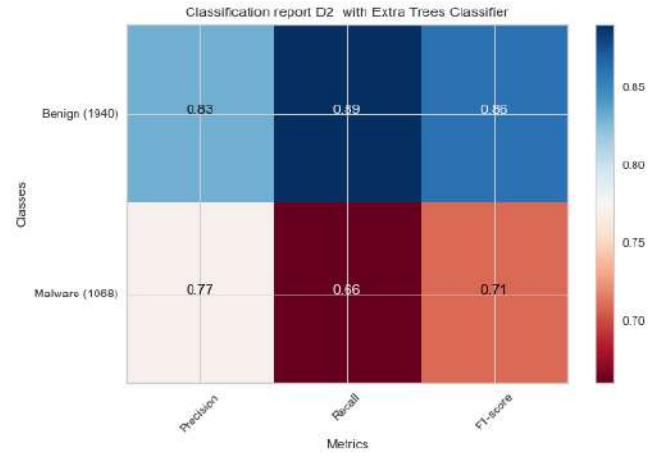


Fig. 9. Classification report D2 with Extra Trees Classifier

our approach the measures of precision, recall and F1 score which vary for both malware and benign classes between 66% and 89% with D_2 . The metrics have differences between the algorithms used by the approach.

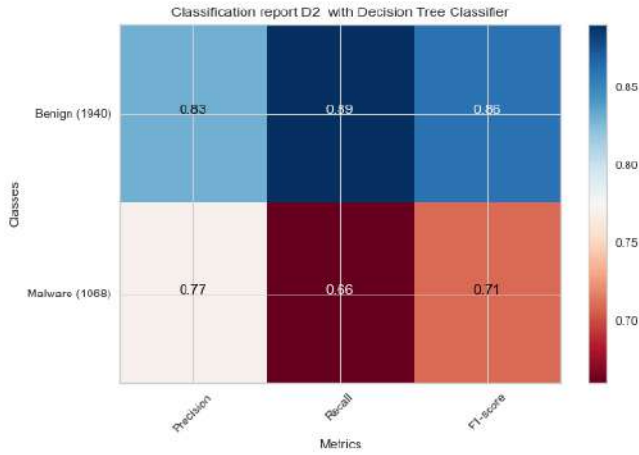


Fig. 8. Classification report D2 with Decision Tree Classifier

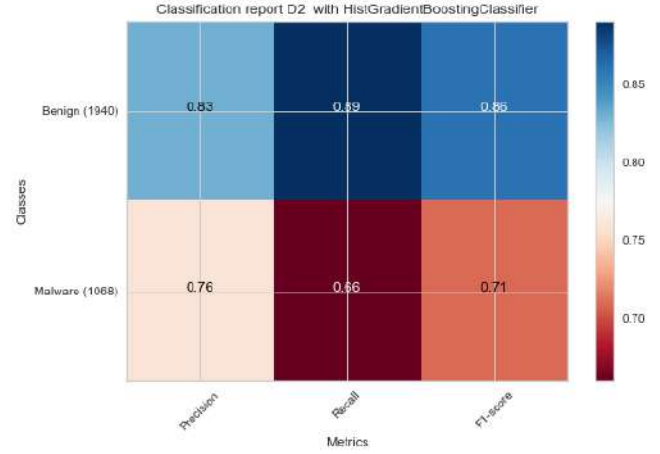


Fig. 10. Classification report D2 with Hist Gradient Boosting Classifier

3) *Results and Analysis with binary classification: Classification report with D3:* Figures 12 - 15 shows the precision, recall and F1 score metrics obtained by UFILA approach applied to the dataset D_4 . The precision, recall and F1 score measures obtained by UFILA vary for two classes between 96% and 99%.

4) *Results and Analysis with binary classification: Classification report with D4:* Figures 16 -19 shows the precision, recall and F1 score metrics obtained by UFILA approach applied to the dataset D_4 . The metrics vary for both classes between 98% and 100% for each of the four used algorithms. We note that for all the algorithms we use, the values of the Precision and F1-score and Recall metrics for the malware class are lower than those for the benign class.

We observed (figure 20) for our four algorithms (Extra

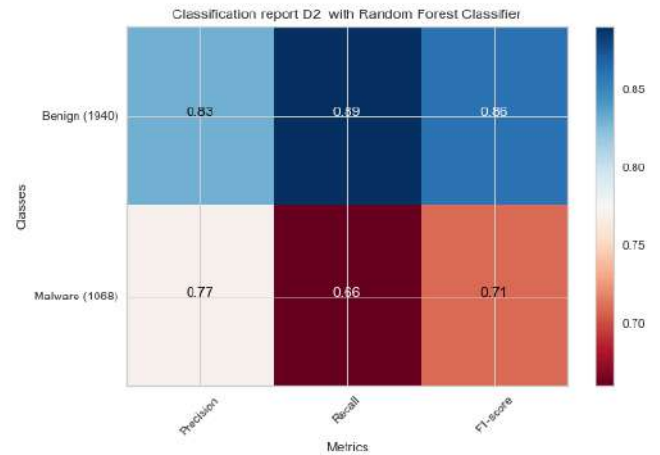


Fig. 11. Classification report D2 with Random Forest Classifier

Trees Classifier, Light Gradient Boosting Machine, Random Forest Classifier, Decision Tree Classifier), that our model

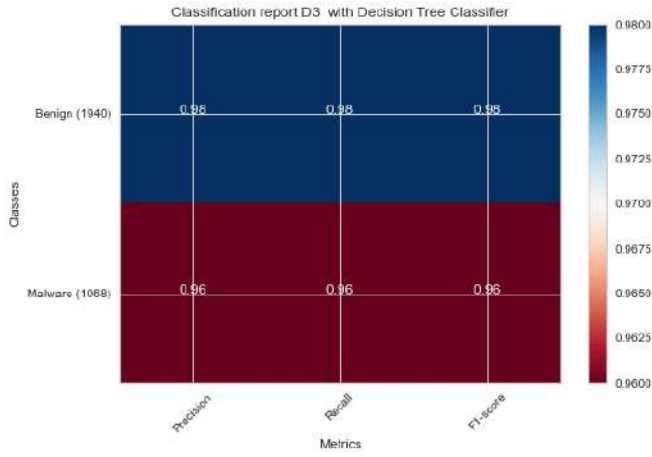


Fig. 12. Classification report D3 with Decision Tree Classifier

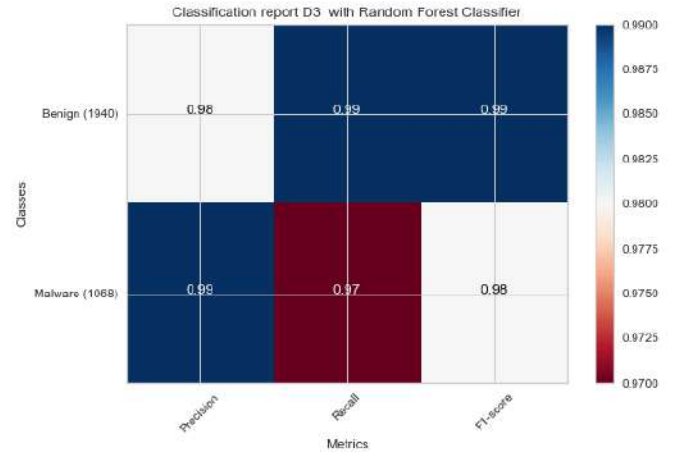


Fig. 15. Classification report D3 with Random Forest Classifier

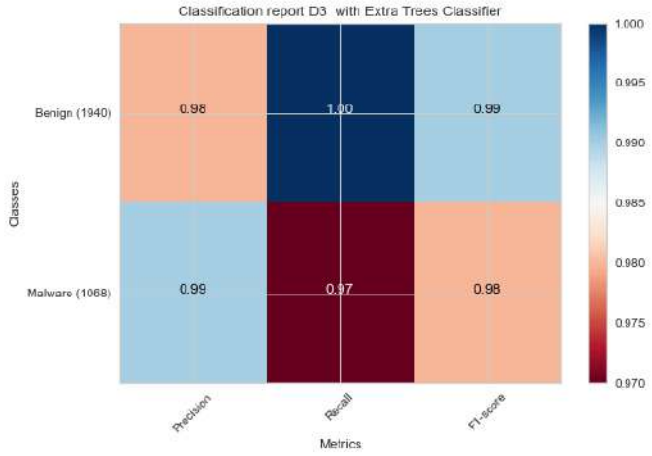


Fig. 13. Classification report D3 with Extra Trees Classifier

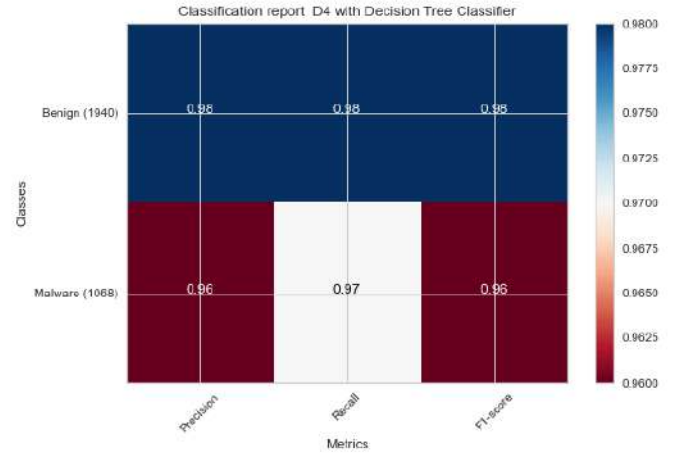


Fig. 16. Classification report D4 with Decision Tree Classifier

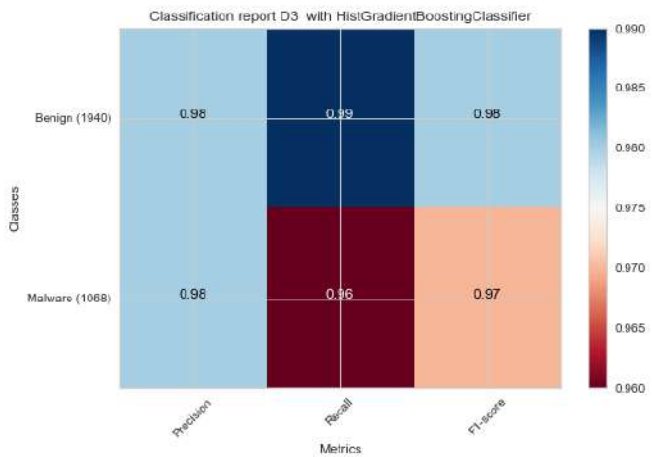


Fig. 14. Classification report D3 with Hist Gradient Boosting Classifier

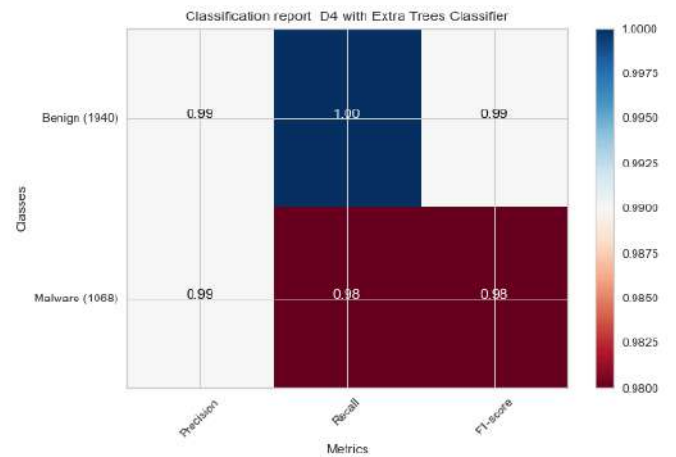


Fig. 17. Classification report D4 with Extra Trees Classifier

becomes more and more precise when going from D_i to D_{i+1} . Thus the precision of the model has increased

considerably from the initial dataset to the final dataset. In addition to the increase in accuracy, the model experiences

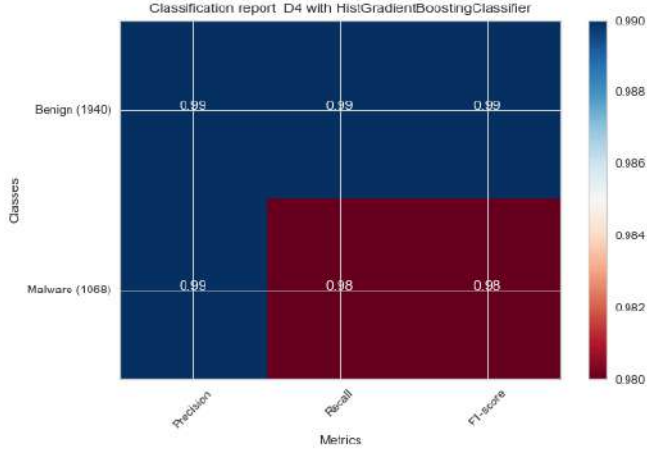


Fig. 18. Classification report D4 with Hist Gradient Boosting Classifier

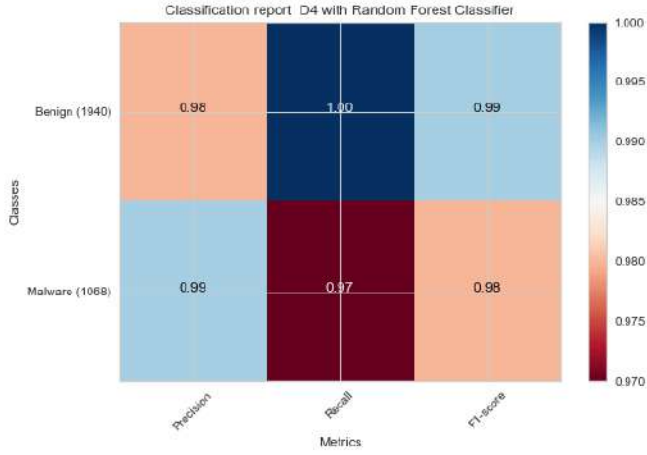


Fig. 19. Classification report D4 with Random Forest Classifier



Fig. 20. Accuracy of UFILA with static features.

an improvement of the accuracy as we add features.

B. Results and Analysis with multi-label classification

In this section, we presented the results obtained after the implementation of our approach. We used dynamic features for this part of our experiment. Our data set is the data collected on the application before simulation and the data collected after simulation. For the needs of our experiment the data has been subdivided into six sub data corresponding to the six categories of features as defined in the section IV-A2

$$D_1=d_1$$

$$D_2=d_2 \text{ Concat } D_1$$

$$D_3=d_3 \text{ Concat } D_2=d_1 \text{ Concat } d_2 \text{ Concat } d_3$$

$$D_4=d_4 \text{ Concat } D_3=d_1 \text{ Concat } d_2 \text{ Concat } d_3 \text{ Concat } d_4$$

$$D_5=d_5 \text{ Concat } D_4=d_1 \text{ Concat } d_2 \text{ Concat } d_3 \text{ Concat } d_4 \text{ Concat } d_5$$

$$D_6=d_6 \text{ Concat } D_5=d_1 \text{ Concat } d_2 \text{ Concat } d_3 \text{ Concat } d_4 \text{ Concat } d_5 \text{ Concat } d_6$$

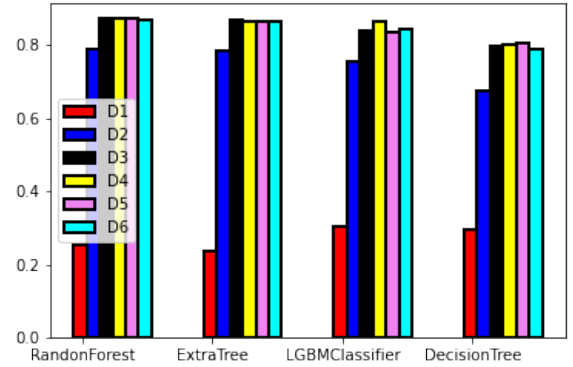


Fig. 21. Accuracy obtained by use dataset before simulation with UFILA approach.

We observed (Figure. 21,22)for our four algorithms (Extra Trees Classifier, Light Gradient Boosting Machine, Random Forest Classifier, Decision Tree Classifier), that our model becomes more and more accurate going from D_i to D_{i+1} with the data before and after simulation.

The results obtained after the implementations show that the accuracy metric has the best performance with the set of data after simulation (Fig.21, Fig.22). Thus, the accuracy of the model has increased considerably between the initial and the final dataset. In addition to the increase in accuracy, the model experiences an improvement in accuracy when we add features. The other metrics such as precision, recall and F-score have the same tendency. We will present the results of using the dataset after simulation with our UFILA approach.

1) *Results and Analysis with multi-label classification with D1* : As can be seen from the figures 23-26, the features in the dataset (D_1) give almost zero values for the different classes at different metrics (precisions, recall, F1-score) With the different classification algorithms our UFILA approach

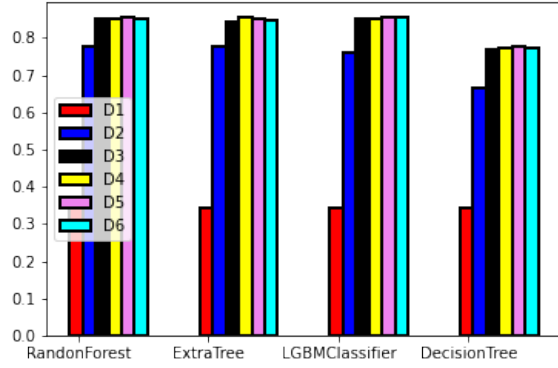


Fig. 22. Accuracy obtained by use dataset after simulation with UFILA approach.

obtains for these two classes the precision of 34%, the recall is 85% and 48% for F1-score.

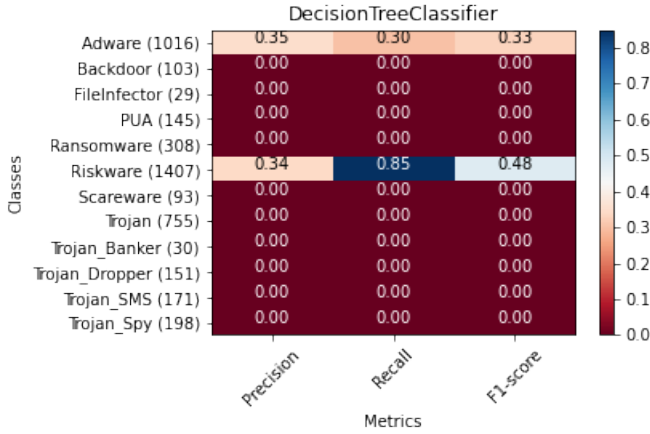


Fig. 23. Classification report D1 with Decision Tree Classifier

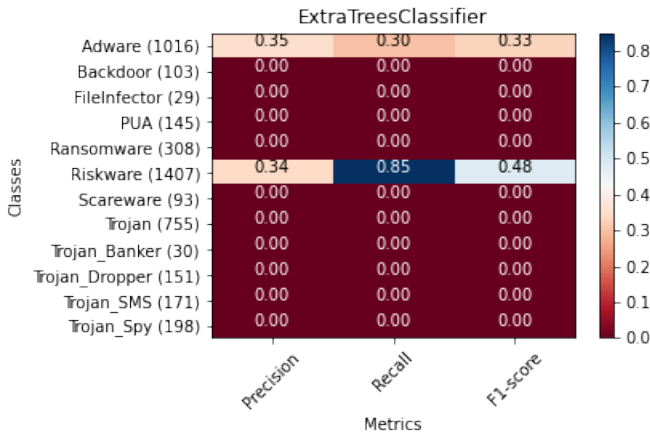


Fig. 24. Classification report D1 with Extra Trees Classifier

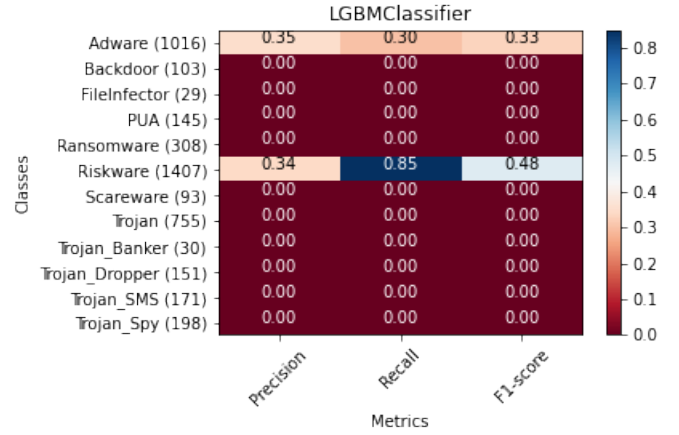


Fig. 25. Classification report D1 with Hist Gradient Boosting Classifier

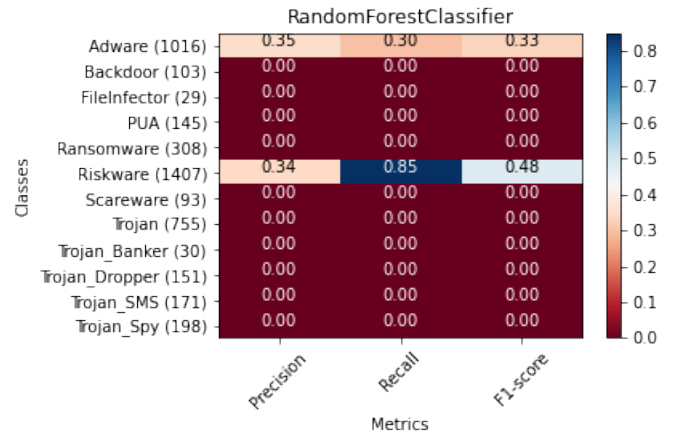


Fig. 26. Classification report D1 with Random Forest Classifier

2) *Results and Analysis with multi-label classification with D2* : We note on figures 27-30 is a considerable improvement in the performance of the metrics of our different models obtained from the algorithms (Extra Trees Classifier, Random Forest Classifier, Light Gradient Boosting Machine Decision Tree Classifier) through our UFILA approach with the addition of the dataset D_2 .

3) *Results and Analysis with multi-label classification with D3* : The figures 31 -34 shows a general improvement in the performance of the metrics of our different models. We can see that the algorithms (Extra Trees Classifier, Light Gradient Boosting Machine) obtain the best performances through our UFILA approach with the added data set D_3 .

C. Classification report with D3

1) *Results and Analysis with multi-label classification with D4* : We have on figures35-38 a improvements on the performance of the metrics of our different models compared to the previous step. We can see that the algorithms (Extra Trees Classifier, Light Gradient Boosting Machine) obtain the best performances thanks to our UFILA approach with the added

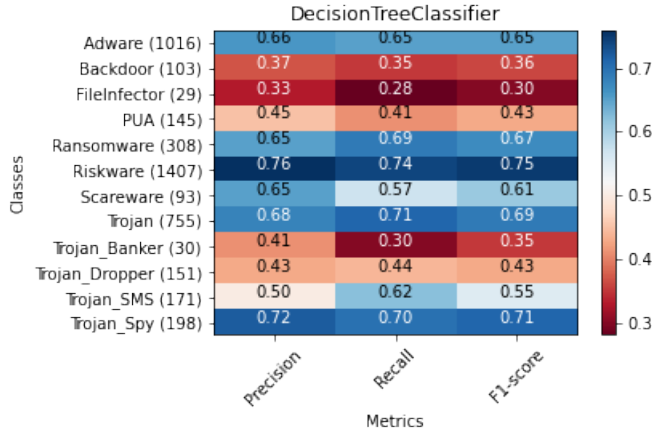


Fig. 27. Classification report D2 with Decision Tree Classifier

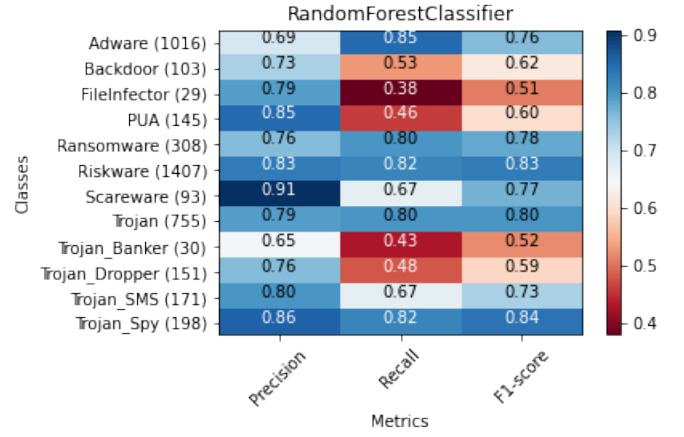


Fig. 30. Classification report D2 with Random Forest Classifier

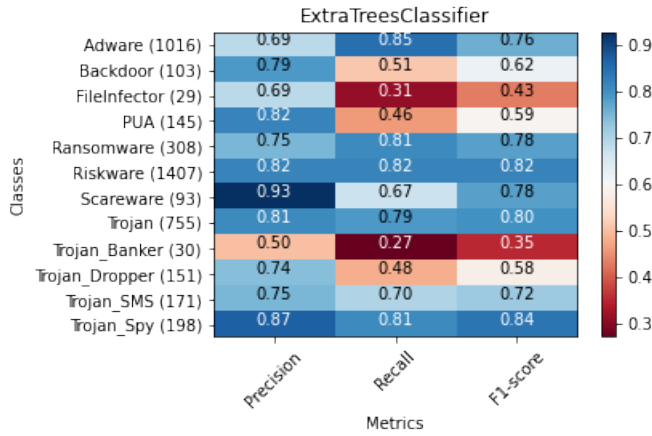


Fig. 28. Classification report D2 with Extra Trees Classifier

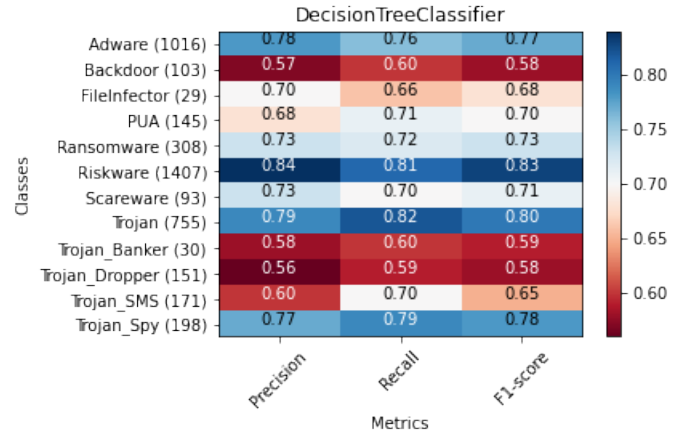


Fig. 31. Classification report D3 with Decision Tree Classifier

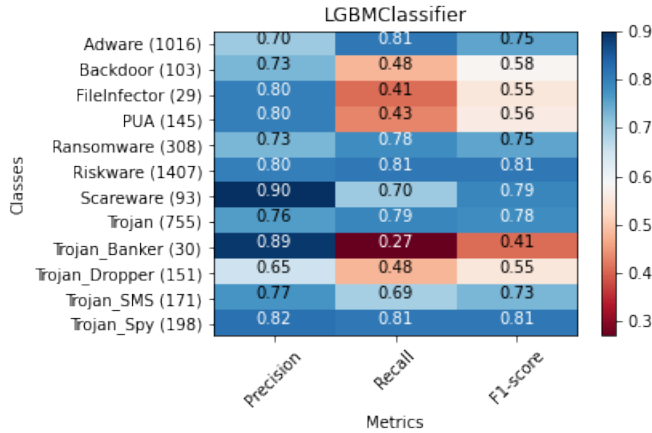


Fig. 29. Classification report D2 with Hist Gradient Boosting Classifier

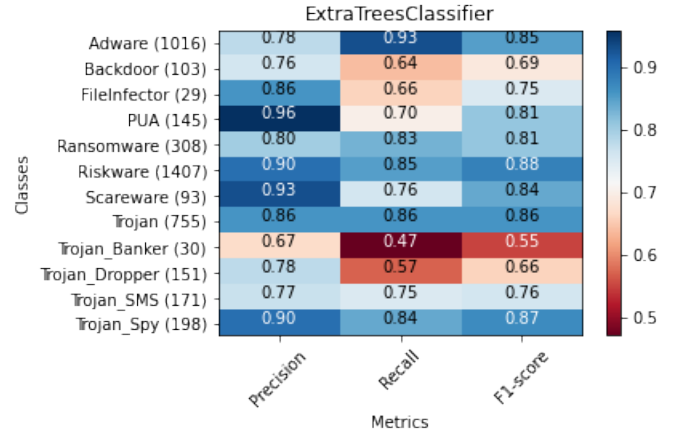


Fig. 32. Classification report D3 with Extra Trees Classifier

dataset D₄. We also see that the classes with few data have low performances.

2) *Results and Analysis with multi-label classification with D₅* : The Figures39 - 39 shows us some improvements

compared to the previous step. We can see that the algorithms (Extra Trees Classifier, Light Gradient Boosting Machine and Random Forest Classifier) obtain the best performances thanks to our UFILA approach with the addition of the D₅ dataset.

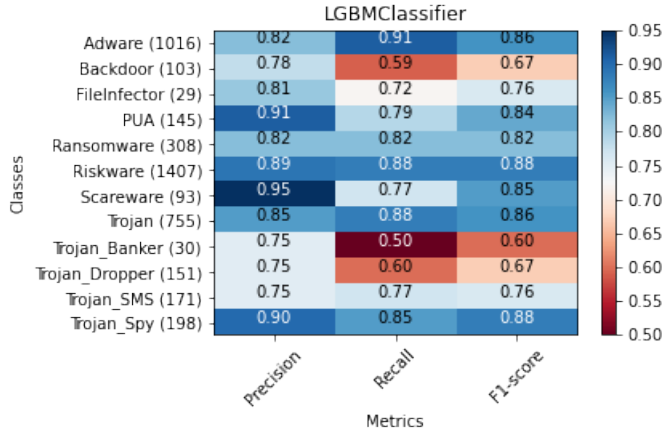


Fig. 33. Classification report D3 with Hist Gradient Boosting Classifier

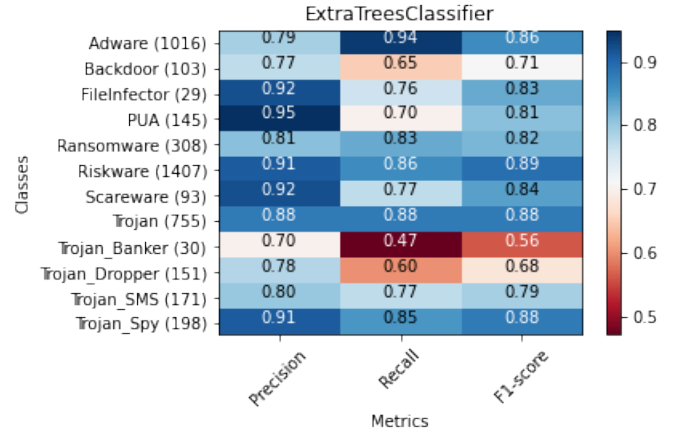


Fig. 36. Classification report D4 with Extra Trees Classifier

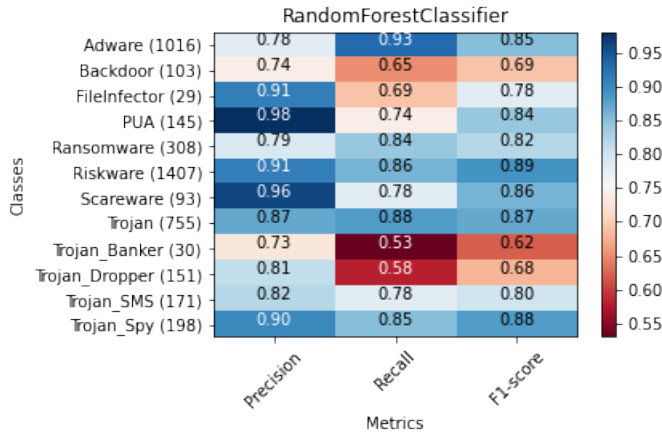


Fig. 34. Classification report D3 with Random Forest Classifier

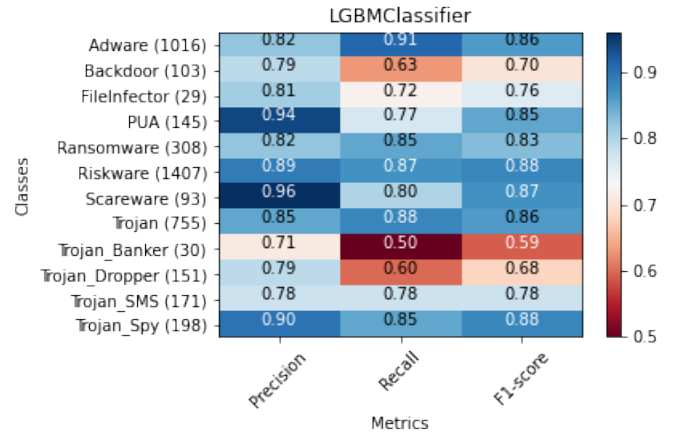


Fig. 37. Classification report D4 with Hist Gradient Boosting Classifier

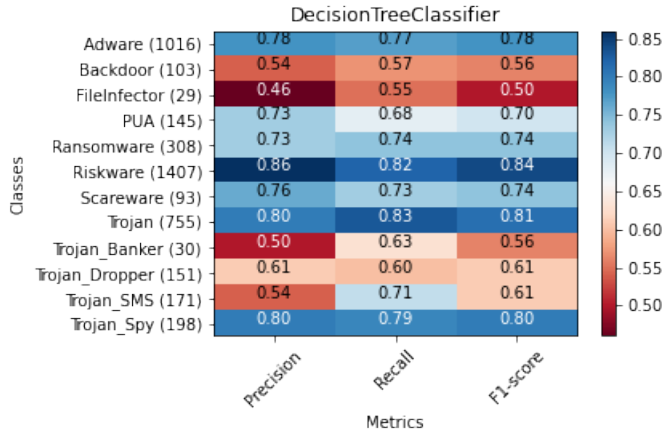


Fig. 35. Classification report D4 with Decision Tree Classifier

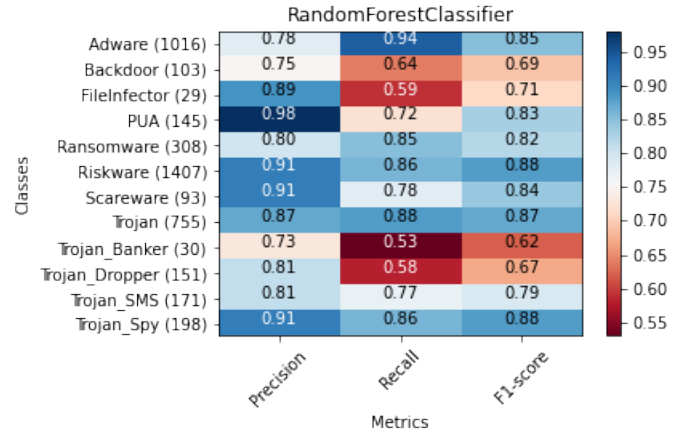


Fig. 38. Classification report D4 with Random Forest Classifier

3) *Results and Analysis with multi-label classification with D6* : The Figures 44 - 46 shows a general improvement in the performance of the metrics of our different models. We can see that the algorithms (Extra Trees Classifier, Light

Gradient Boosting Machine and Random Forest Classifier) obtain the best performances through our UFILA approach with the added data set D₆.

We observed (Fig.44 - 46) that the different metrics (pre-

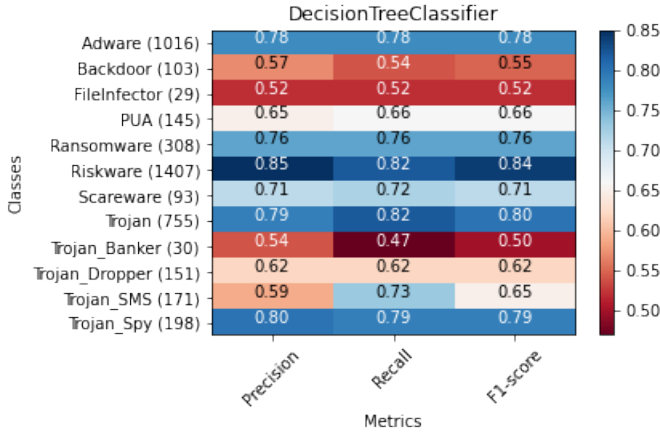


Fig. 39. Classification report D5 with Decision Tree Classifier

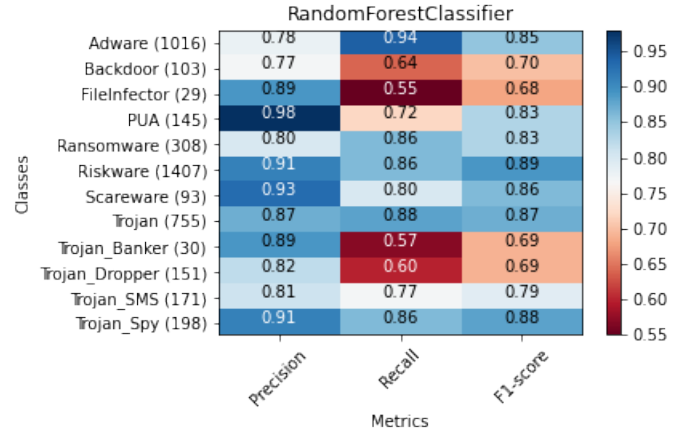


Fig. 42. Classification report D5 with Random Forest Classifier

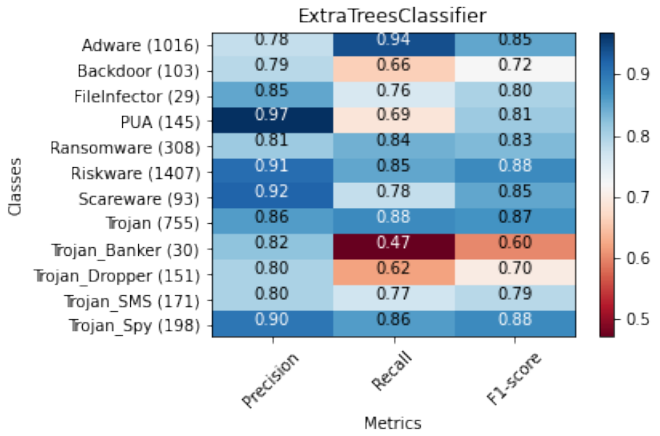


Fig. 40. Classification report D5 with Extra Trees Classifier

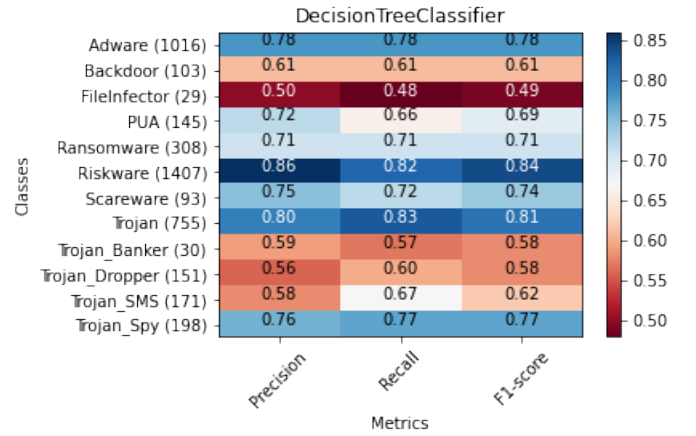


Fig. 43. Classification report D6 with Decision Tree Classifier

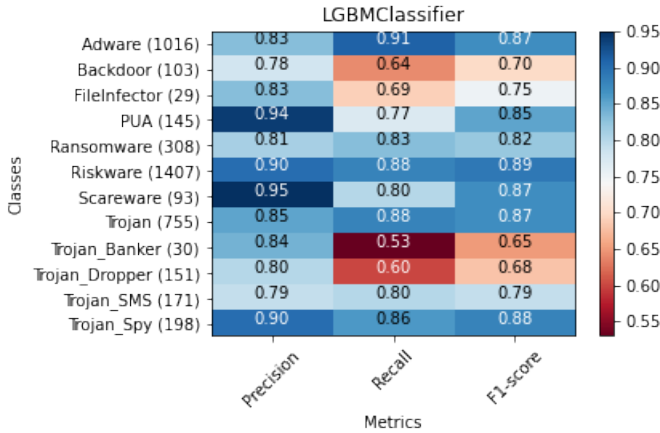


Fig. 41. Classification report D5 with Hist Gradient Boosting Classifier

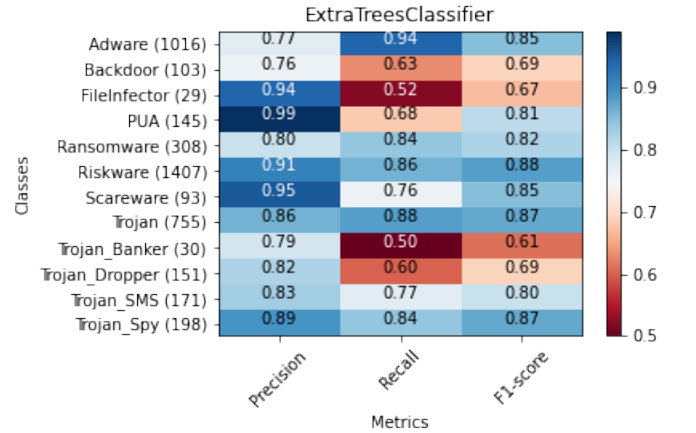


Fig. 44. Classification report D6 with Extra Trees Classifier

cision, recall, F1_score) obtained from our approach have considerably increased between the initial and the final dataset. Thus our approach becomes more and more efficient as we add more features. That is, as we go from D_i to D_{i+1} .

Our approach is highly and validly relevant since it achieves highly appreciated performances for the different metrics (Accuracy, Recall, Precision and F1-score), and this with static and dynamic characteristics. Through our approach we have

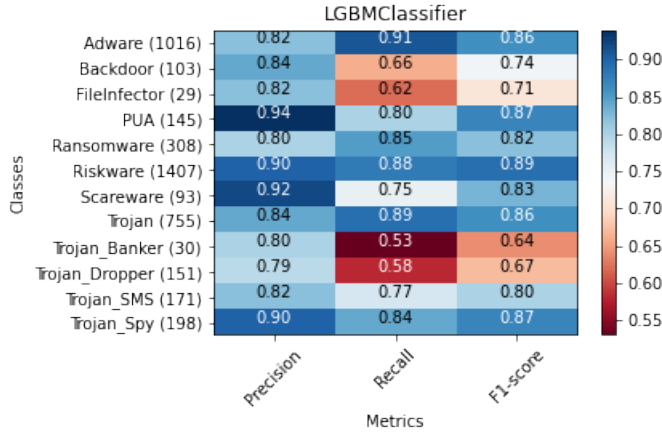


Fig. 45. Classification report D6 with Hist Gradient Boosting Classifier

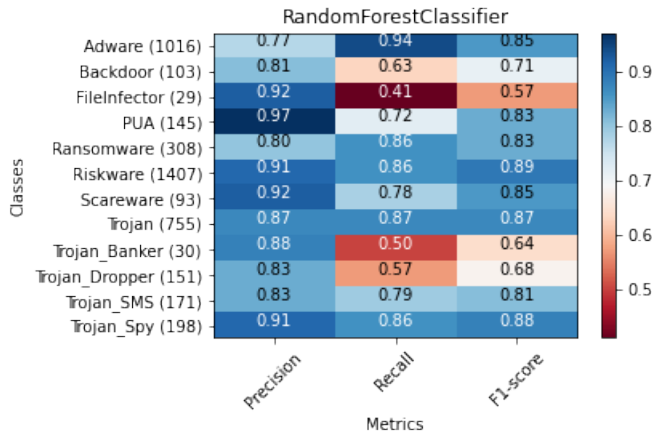


Fig. 46. Classification report D6 with Random Forest Classifier

identified three algorithms (Light Gradient Boosting Machine, Extra Trees Classifier and Random forest Classifier) that allow us to obtain satisfactory results. We also note that the performance of the classification metrics precision, recall and F1-score are 99%, 98.6% and 98% respectively for the static and binary datasets. For the multi-class and dynamic datasets, the performance of these same metrics varies between 80% and 92% depending on the class and the type of algorithm used. This enables us to say that the model built from our UFILA approach achieves very good performance values in terms of classification while updating features. With the UFILA approach, the classification algorithms outperform the results obtained in classical machine learning (Fig.8) and in DroifFusion.

VI. CONCLUSION

In this paper, we proposed an incremental learning approach (UFILA) with the general objective of Android malware detection and classification. In our proposal, we use basic classifiers and a feature update function. This function allows new features to be added. We evaluated UFILA first with a

binary and static dataset and second with a multi-class and dynamic dataset. For the binary and static dataset we used split the features into four distinct types from the same malware and benign dataset. The results presented demonstrate its effectiveness in improving performance. We obtained respective performances for Accuracy, Precision, Recall and F1-score 99%, 99%, 98.6%, 98%, 98%. For multi-class and dynamic datasets, the features were split into six distinct types from the same malware and benign dataset. We obtained performances that vary between 80% and 92% depending on the class and the type of algorithm used for the metrics (Accuracy, Precision, Recall and F1-score). The approach we have composed allows to detect malicious android applications from static and dynamic features with an update of the features. It also allows to effectively classify the different classes. Our approach outperforms many literature-based automatic malware detection approaches [11] [30]. In our future work, we plan to better realize and optimize the update function.

Acknowledgment

This work was supported by The Partnership for skills in Applied Sciences, Engineering and Technology - Regional Scholarship and Innovation Fund (PASET-RSIF).

REFERENCES

- 1 Digital 2022: Another year of bumper growth. Published by Joseph Johnson, Jan 26, 2022 <https://wearesocial.com/fr/blog/2021/01/digital-report-2021/>. Accessed: 2022-04-30.
- 2 Digital 2021: 3.8 billion people use social media. <https://www.statista.com/statistics/325201/countries-share-of-malicious-attacks/>. Accessed: 2021-05-24.
- 3 Juan Contreras, Ruth Cano, and José García. *Applied Informatics*, volume 942. Springer International Publishing, 2018.
- 4 Jae-wook Jang, Hyunjae Kang, Jiyoung Woo, Aziz Mohaisen, and Huy Kang Kim. Andro-Dumpsys : Anti-malware system based on the similarity of malware creator and malware centric information Andro-Dumpsys : Anti-malware system based on the similarity of malware creator and malware centric information. 58(January 2019):125–138, 2016.
- 5 Jun Song, Chunling Han, Kaixin Wang, Jian Zhao, Rajiv Ranjan, Jun Song, Chunling Han, Kaixin Wang, Jian Zhao, Rajiv Ranjan, and Lizhe Wang. An integrated static detection and analysis framework for android An Integrated Static Detection and Analysis Framework for Android. *Pervasive and Mobile Computing*, 2016.
- 6 Vikas Sihag, Manu Vardhan, and Pradeep Singh. A survey of android application and malware hardening. *Computer Science Review*, 39:100365, 2021.
- 7 Kewen Zou, Xi Luo, Pengfei Liu, Weiping Wang, and Haodong Wang. *ByteDroid: Android Malware Detection Using Deep Learning on Bytecode Sequences*, volume 1149 CCIS. Springer Singapore, 2020.
- 8 Danish Vasan, Mamoun Alazab, Sobia Wassan, Hamad Naeem, Babak Safaei, and Qin Zheng. IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture. *Computer Networks*, 171(April 2019):107138, 2020.
- 9 Vijaypal Singh Rathor, Bharat Garg, and G K Sharma. New lightweight Anti-SAT block design and obfuscation technique to thwart removal attack. *Integration*, 75:178–188, 2020.
- 10 Anukriti Sinha. Emulation vs Instrumentation for Android Malware Detection. 2019.
- 11 Gopi Sanghani and Ketan Kotecha. Incremental personalized E-mail spam filter using novel TFDCR feature selection with dynamic feature update. *Expert Systems with Applications*, 115:287–299, 2019.
- 12 Zakaria Sawadogo, Gervais Mendy, Jean Marie Dembele, and Samuel Ouya. Android malware classification: Updating features through incremental learning approach(ufila). In *2022 24th International Conference on Advanced Communication Technology (ICACT)*, pages 544–550, 2022.

- 13 Dongfang Li, Zhaoguo Wang, Lixin Li, Zhihua Wang, Yucheng Wang, and Yibo Xue. FgDetector : Fine-grained Android Malware Detection. pages 311–318, 2017.
- 14 Linfeng Wei, Weiqi Luo, Jian Weng, Yanjun Zhong, Xiaoqian Zhang, and Zheng Yan. Machine Learning-Based Malicious Application Detection of Android. 5:25591–25601, 2017.
- 15 Guangyu Wang Liu and Zhijing. *Android Malware Detection Model Based on LightGBM*. 2018.
- 16 Michal Kedziora, Paulina Gawin, and Michal Szczepanik. MALWARE DETECTION USING MACHINE LEARNING ALGORITHMS AND REVERSE ENGINEERING OF ANDROID J AVA CODE. 11(1):1–14, 2019.
- 17 Mohammad Modallal. Android Malware Detection Using Machine Learning with User Feedback and Static Features. 2020.
- 18 Moutaz Alazab, Mamoun Alazab, Andrii Shalaginov, and Abdelwaddood Mesleh. Intelligent mobile malware detection using permission requests and API calls. *Future Generation Computer Systems*, 107:509–521, 2020.
- 19 Yanyun Lu, Khaled Boukharouba, Jacques Boonert, Anthony Fleury, and Stéphane Lécœur. Application of an incremental SVM algorithm for on-line human recognition from video surveillance using texture and color features. *Neurocomputing*, 126:132–140, 2014.
- 20 Alexandre R. Mello, Marcelo R. Stemmer, and Alessandro L. Koerich. Incremental and decremental fuzzy bounded twin support vector machine. *Information Sciences*, 526:20–38, 2020.
- 21 Clem Brooks and Jeff Manza. M Ethods F or I ncremental L Earning : a S Urvey. 3(4):119–125, 2013.
- 22 Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D. Bagdanov, and Joost van de Weijer. Class-incremental learning: survey and performance evaluation. pages 1–24, 2020.
- 23 Junwei Zhong, Zhenyan Liu, Yifei Zeng, Lijia Cui, and Zizheng Ji. A Survey on Incremental Learning. (Cape):166–174, 2017.
- 24 Aida Chefrour. Incremental supervised learning: algorithms and applications in pattern recognition. *Evolutionary Intelligence*, 12(2):97–112, 2019.
- 25 Zhetao Li, Wenlin Li, Fuyuan Lin, Yi Sun, Min Yang, Yuan Zhang, and Zhibo Wang. Hybrid malware detection approach with feedback-directed machine learning. 63(March):1–3, 2020.
- 26 Eric Lopez-Lopez, Carlos V. Regueiro, and Xose M. Pardo. Incremental learning from low-labelled stream data in open-set video face recognition. *arXiv*, 2020.
- 27 A L Sangal. DeepDroid : Feature Selection approach to detect Android malware using Deep Learning. pages 19–22, 2019.
- 28 Hamza Erol, Bala Mikat Tyoden, and Recep Erol. Classification Performances of Data Mining Clustering Algorithms for Remotely Sensed Multispectral Image Data. *2018 IEEE (SMC) International Conference on Innovations in Intelligent Systems and Applications, INISTA 2018*, pages 1–4, 2018.
- 29 Daniel Arp, Michael Spreitzenbarth, H Malte, Hugo Gascon, and Konrad Rieck. DREBIN : Effective and Explainable Detection of Android Malware in Your Pocket.
- 30 Suleiman Y. Yerima and Sakir Sezer. DroidFusion: A Novel Multilevel Classifier Fusion Approach for Android Malware Detection. *IEEE Transactions on Cybernetics*, 49(2):453–466, 2019.
- 31 AndMal 2020 — Datasets — Research — Canadian Institute for Cybersecurity — UNB.
- 32 David M. W. Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *CoRR*, abs/2010.16061, 2020.



Cheikh Anta Diop of Dakar and the Laboratoire d'Analyse

Zakaria SAWADOGO received his Master II degree in software engineering and information systems from Joseph Ki-Zerbo University in Burkina Faso. He is currently a researcher in cybersecurity including artificial intelligence and is affiliated with the Laboratoire d'Informatique, de Télécommunications et Applications (LITA) of the University

Numérique et d'Informatique (LANI) of the University Gaston Berger of Saint Louis. His research interests include mobile security, system security and artificial intelligence. He is a member of the IEEE.



Things (IoT), Artificial Intelligence and IT Security. He is a member of the LITA Laboratory of UCAD.



intelligence, agent-based modeling, dynamical systems, evolutionary algorithms, genetic regulatory networks, machine learning



Pr. Samuel Ouya is currently the director of the LITA laboratory at the UCAD. He was from 2013 to May 2017 the first Director of Infrastructure and Information System of the first virtual university of Senegal (UVS). Holder of a thesis in Applied Mathematics from the Gaston Berger University of Saint-Louis Senegal and a Telecommunications Thesis from the UCAD university in Dakar-Senegal, he is interested in he is interested in Applications of innovative telecom services to virtual organizations.