DIGITAL FORENSIC RESEARCH CONFERENCE

Machine Learning Based Approach to Analyze File Meta Data for Smart Phone File Triage

By:
Cezar Serhal (University College Dublin)
and Nhien-An Le-Khac (University College Dublin)

From the proceedings of

The Digital Forensic Research Conference

DFRWS USA 2021

July 12-15, 2021

Forensic Science International: Digital Investigation 37 (2021) 301194

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

journal homepage: www.elsevier.com/locate/fsidi

DFRWS 2021 USA - Proceedings of the Twenty First Annual DFRWS USA

Machine learning based approach to analyze file meta data for smart phone file triage

Cezar Serhal, Nhien-An Le-Khac*

University College Dublin, Belfield, Dublin 4, Ireland

ARTICLE INFO

Article history:

Keywords:
Smartphone file triage
Mobile forensic triage
Metadata
Classification
Random forest
Neural networks
Mobile phone forensics

ABSTRACT

With the rapid increase in mobile phone storage capacity and penetration, digital forensic investigators face a significant challenge in quickly identifying relevant examinable files within a plethora of uninteresting OS and application files extracted by forensic tools. This challenge can have serious adverse effects in time critical cases, and can also result in increasing case backlog. A possible solution for this issue is to prioritize digital artifacts. This is referred to as triage. Several digital forensic triage methodologies based on classical automation techniques such as block hash and regular expression matching have been proposed. However, such techniques suffer from the significant limitation of requiring users to know and hardcode data templates and relations of interest. In literature, more flexible machine learning based approaches have been proposed to classify whether a mobile device, rather than a mobile device artifact, is of interest or not based on its usage metrics and file-system metadata. Also, recently an approach has been proposed and tested in triaging data generated and extracted from a computer-based operating system. However, this approach did not cover smart mobile operating system, and it did not consider key steps such as feature engineering, feature selection, and hyper-parameter tuning. Hence, in this paper, we propose a comprehensive machine learning based solutions with features extracted from file metadata to identify possible smart phone files of interest that should be examined. A range of classification algorithms are tested and their performance compared. Our classification models were trained and tested on a dataset consisting of the metadata of nearly 2 million files extracted from devices running Android OS and linked to real terrorism cases. The use of real case data allows obtaining realistic results, and restricting the operating system and case type helps narrow the experimentation scope enough to provide a proof of concept. Through our experiments, a best classifier is also identified.

1. Introduction

With average mobile phone penetration in developed and developing countries exceeding 90% (Deloitte, 2017), Law enforcement (LE) agencies are faced with an increscent number of cases linked to mobile phones (Marturana et al., 2011b; Faheem et al., 2014). Coupled with the increasing storage capacity of mobile phones which reached an average of 82.9 GBs in 2019 (LIM, 2019), digital forensic investigators face a significant challenge in quickly identifying relevant examinable files within a plethora of uninteresting OS and application files extracted by forensic tools.

Existing extraction and examination tools focus on finding all the evidence (Gómez, 2012; Witteman et al., 2016). This results in presenting examiners with a large number of files most of which have no added value for the investigation. For example, when considering the dataset used for this research which consists of 1,998,950 files extracted from 12 devices, only 6% of the files were of interest for the examiners. Thus, manually examining all extracted files takes up valuable investigation time and occupies skilled resources in examining file of no relevance. This puts additional pressure on examiners and forensic labs where it is common to have a 9–12 months of backlog (Gómez, 2012; Hitchcock et al., 2016). In addition, this approach is ineffective in time critical cases where examiners need to identify evidence in the shortest time possible.

To tackle this issue, triage techniques can be used to classify files based on their possible interest. More specifically, automated tools that embed the examiners' knowledge and skills can be used to perform this task. Several digital forensic triage methodologies based on classical automation techniques such as block hash and

* Corresponding author.
E-mail address: an.lekhac@ucd.ie (N.-A. Le-Khac).

https://doi.org/10.1016/j.fsidi.2021.301194

regular expression matching have been proposed. For instance, bulk_extractor presented in (Garfinkel, 2013). However, such techniques suffer from the significant limitation of requiring LE users to know and hardcode data templates and relations of interest. For example, a dictionary of known file hashes can be used to identify generic files. Yet, if a single bit is modified due to an update or due to user activity, the files will no longer be identified as not of interest.

Recently, more flexible machine learning (ML) based approaches have been proposed to classify whether a device is of interest based on its usage metrics and file-system metadata (Dalins et al., 2018; Gómez, 2012; Marturana et al., 2011a). Due to the difficulty of obtaining train and test data related to criminal cases, experiments undertaken by existing research are limited to a dataset related to intellectual property or child pornography made available to the researches by an Italian LE agency. Researchers tried several ML classification algorithms to address this problem based on phone characteristics and usage data such as the phone type, number of contacts, number of received calls, etc. Other researchers applied ML on file and system metadata to identify the owner of carved data or to select relevant events to construct a cybercrime events timeline. Only one recent research (Du and Scanlon, 2019) proposed a ML approach to classify files based on metadata. However, the results are based on data generated and extracted from a computer-based operating system rather than a smart mobile operating system. In addition, it did not consider key steps such as feature selection and hyperparameter tuning.

This paper aims at answering the question of whether file metadata and ML can be used to decide if a file extracted from a smart phone should be examined or not. The research question is built upon the hypothesis that file metadata can indicate the relevance of a file for an investigation, and that ML classification algorithms can model the decision-making process required to identify files of interest. In addition, it is hypothesized that different classification models will perform differently. Therefore, this effort presents an approach that addresses the research question, tests different ML classification models, and compares their performance on a real-world dataset. The proposed approach includes:

- Forensically extracting available file metadata from mobile devices.
- Engineering features, and applying a feature selection approach to select the subset of features that is expected to maximize performance.
- Designing different ML classification models with tuned hyper-parameters.
- Using cross-validation and real-world datasets to evaluate and compare the performance of the different ML models.

To limit the scope of experimentation, mobile phones related to terrorism cases and running Android operating system are used. However, the same approach can be applied on data extracted from smart phones running other types of operating systems and related to other types of crime.

This research effort presents a robust approach to triage files extracted from a smart phone and can be considered a proof of concept to develop ML based triage tools for files extracted from smart phones.

The rest of this paper is organized as follows: Section 2 presents related work on ML approaches for digital triage and for analyzing file metadata in Digital Forensics. Section 3 describes the adopted approach. The conducted experiments and results are illustrated in Section 4. Section 5 discusses and evaluates the obtained results. Finally, Sections 6 concludes this paper.

2. Related work

2.1. Machine learning approaches for digital triage

With the ultimate goal of decreasing forensic lab backlog (Gómez, 2012), introduced a triage tool that attempts to identify if an examined device is relevant to a specific crime by using different ML classification algorithms such as Naïve Bayes (NB), Decision Tree (DT), K-Nearest Neighbor (KNN) and Support Vector Machines (SVM) (Sester et al., 2021). The paper takes child pornography and intellectual property theft as two examples of crimes. Several classification algorithms are used and their results compared. Through their experiments with 21 forensic images of hard disk, the best performance is achieved by KNN reporting an accuracy of 90%. Their approach focused on identifying if a device is important for analysis based on device usage information.

(Marturana et al., 2011a) presented a quantitative approach to mobile forensic in-lab triage using ML. The approach is based on the work in (Marturana et al., 2011b), which is a similar study that processes reports and data extracted from mobile devices seized by the Italian Cybercrime LE. Several classification algorithms such as Bayes Net and DT are used to determine the likelihood that a device was used to commit a crime related to pedophilia. Their approach aimed at classifying if a device rather than a file is relevant for an investigation.

(Du and Scanlon, 2019) presented a ML based approach for automated identification of incriminating digital forensic artifacts based on file metadata. However, the experimentation is limited to files extracted from computers, and not from smart mobile phones. In addition, a limited dataset generated by the researchers is used in the experiments. The model is designed with 9 features, 4 of which refer to time. The file extension is the only categorical feature used, and neither feature selection nor model hyper-parameter tuning was applied in the approach. Their research effort showed that performance is affected by the prevalence of the class of interest. This highlights the importance of using data similar to real world cases in order to generate classifiers that are effective in practice.

(Dalins et al., 2018) designed and tested a deep-learning based child exploitation material classifiers based on the content of pictures using pornography and real case data from the Australian Federal Police. Although the taken approach utilizes content rather than metadata, they presented a classifier sufficient for triage and early warning for existence of child exploitation material. However, the classifiers performance is problematic when it comes to correctly classifying the severity of the images against existing scales.

2.2. Applications of ML in analyzing file metadata in digital forensics

The use of ML on file metadata in digital forensics is not restricted to the triage field. It is also used to answer other similar question including ones that in essence are classification problems.

(Garfinkel et al., 2011) presented a solution to identifying the owner of data carved from storage media used by multiple users. Classification algorithms such as KNN and DT are used in conjunction with file system metadata and extended file metadata to calculate the ownership likelihood for each possible user. They found that DT performed better than the tried KNN algorithms.

(Mohammad, 2019) used classification algorithms to reconstruct cybercrime events based on file system metadata. They compared the performance of five ML classification algorithms (NB, SVM, Random Forests (RF), Regression Trees (RT) and Neural Network (NN)) in identifying relevant files based on file system

metadata. They found that the NN and the RF algorithms achieved the highest precision of 89%.

(Milosevic et al., 2017) implemented two approaches based on ML algorithms including SVM with Sequential Minimal Optimization, NB, DT, JRIP, and logistic regression to detect Android malware. Ensemble learning (using AdaBoost), which combines the results of multiple algorithms, was used to improve performance.

(Khan and Wakeman, 2006) used Recurrent Neural Networks to reconstruct a post-event timeline that in essence identifies which and when applications where run. The approach is based on monitoring file system changes that occur when specific programs are executed. Internet explorer was selected as the use case to demonstrate proposed approach. Accuracy of the implemented algorithm increased with increasing the duration of file system activity of the training data. Key limitations of the approach include the need for training a separate neural network for each application or a significantly different version of an application.

2.3. Literature review conclusion

Surveyed literature shows that classical automation of triage includes techniques such as block hash and regular expression matching. The main drawback of such techniques is the requirement to know and hardcode templates and relations for the data of interest, while in many cases such information is difficult to theoretically discover. More flexible and advanced techniques relying on ML were used to mainly classify whether a device was of interest or not based on its usage metrics and file-system metadata. Most of the ML approaches used similar model design concepts such as the use of feature normalization, vectorizing non-numeric features, and 10-fold cross validation. Some experiments used all features, other reduced the features manually or automatically.

Also, file metadata in conjunction with ML techniques were used to solve other classification problems in digital forensics such as detection of malware or identification of carved data owner. Only one recent paper tackled the issue of classifying files based on their metadata. However, the experimentation was limited to files generated from a computer operating systems, and the presented approach did not cover key areas such as feature selection and hyper-parameter tuning.

A significant portion of the extracted smart phone files usually consists of generic operating system and application files that have no added value for the investigation. Therefore, digital forensic investigators are facing increasing challenges in quickly identifying files of interest dispersed among uninteresting files.

3. Methodology

This section describes the experimentation scope, corpus, utilized tools, and adopted approach. In addition, this section describes key steps in the approach such as, preprocessing, feature selection, hyper-parameter tuning, and utilized performance metrics.

3.1. Scope

In order to narrow the experimentation scope, this research will only consider:

- Android OS devices: Android OS devices were chosen as they currently have a smart OS market share exceeding 70% (StatCounter, 2020).
- Terrorism related cases: The models will be trained and tested using data extracted from real cases. This topic was selected due to its current importance for the law enforcement community.

In addition, the focus is on in-lab digital forensic triage techniques opposed to on-site techniques.

3.2. Corpus

The corpus consists of the metadata of around 2 million files extracted from 12 devices running Android OS and whose users are linked to terrorism cases. As this is real data, access to it was very restricted and limited to a controlled environment.

3.3. Tools and platforms

In terms of software, Python (version 3.7) was selected as the programming language for metadata extraction, pre-processing and model development and testing. Among the key libraries used are: Scikit-Learn (Version 0.22.1), Pandas (Version 1.0.1), Seaborn (version 0.10.0) and Matplotlib (version 3.1.3). XRY (Version 8.2.4) and XRY Reader (Version 6.28) from MSAB ("MSAB Products," n. d.) are used for imaging and extracting files from the target phones. In terms of hardware, training and testing was conducted on a workstation with the following specifications: MS Windows 10 (64 bit) with an Intel Core i7-8700 CPU @3.2–4.6 GHz (6 Cores - 12 Threads) and 32 GBs DDR4 RAM.

3.4. ML models

The following range of classifiers are tested in our approach: NB, SVM, KNN, DT namely Classification and Regression Trees (CART), Random Forests (RF) and Neural Network – Multi Layer Perceptron (NN-MLP).

3.5. Approach

A forensically sound approach that does not affect the evidence is used to conduct in-lab triage for files extracted from smart phones using different ML classification algorithms. The adopted approach consists of the following key steps: (1) forensically extracting files from mobile phones; (2) pre-processing; (3) feature selection, (4) tuning the hyper parameters of the tested classification algorithms, (5) training and testing the performance of the algorithms with 10-fold cross validation, and (6) evaluating the performance of the different algorithms.

The approach starts by obtaining logical images of the devices made available for this experiment. Next, XRY Reader from MSAB is

Fig. 1. Sample folder structure for extracted files.

used to extract files from the obtained images. Extraction results in (a) files organized in folders based on file type (Audio, Documents, Pictures, and Videos) as shown by the folder tree in Fig. 1; (b) extraction log files that contain all extractable file metadata. Fig. 2 illustrates an example of a log file entry for a picture file that includes the metadata such as File Name, Path, File Type, File Size, Modified Date, SHA1 Hash, X-Resolution, Y-Resolution and EXIF data (if available). Fig. 3 shows sample log file entries for a video file and a document.

3.6. Pre-processing

Pre-processing includes exploring and labeling data, creating features, cleaning data, encoding categorical data, and normalizing features.

To label the data, extracted files are manually examined by forensic experts and then assigned to one of the following two classes 'Interesting' or 'Not Interesting'. A file is considered of interest if (a) its content should be examined and (b) if it has comprehensible data. For example, an uninteresting file could also include a non-generic yet very small and incomprehensible thumbnail, or an empty audio file. This step resulted in labeling 94% of extracted files as 'Not Interesting'.

A data cleansing strategy is also adopted where records are checked for missing data and data variations. Variations are normalized (e.g., file size units), and missing data is imputed if possible or dropped.

Textual data records such as file path are tokenized and encoded. Then, data is normalized by using Min-MaxScaler from the Scikit Learn library.

This step results in a corpus of 1,998,950 files ready for experimentation.

3.7. Feature selection

Many features and feature subsets can be proposed for the ML models. In addition, the feature space can significantly grow after including and encoding categorical features such as path and extension. Training and testing a large number of irrelevant features on a large dataset could significantly affect the execution time and predictive performance of the model. Hence, one of most challenging tasks is to identify the optimal set of metadata features.

We first consider all available metadata to create a range of features, then feature selection is used to select the best feature set. Feature creation is influenced by field expertise (Bertè et al., 2012)

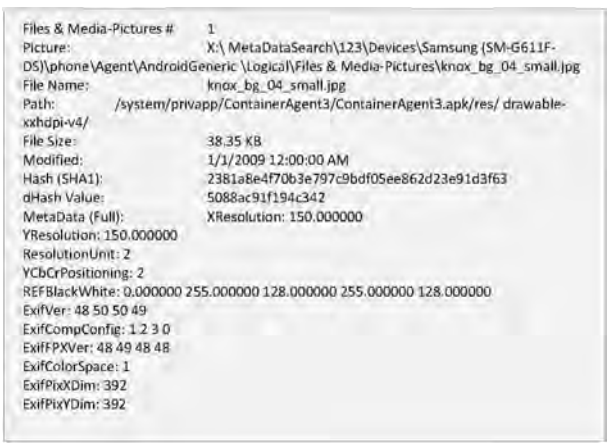

Fig. 2. Sample XRY Log File entry of a picture file.

which helps produce relevant features that also take into consideration the examiners line of thought when determining the interest of a file.

Accordingly, the following 12 base features are created:

- General:
  1. File Type (extension)
  2. File Size (KB)
  3. Delta Apprehended (Days) (Apprehended date - file modified date)
  4. EXIF Flag (existence of EXIF data)
  5. Xresolution (Photos only)
  6. Yresolution (Photos only)
- Filename related:
  7. Number of characters in a name
  8. % of numbers in name
  9. % of underscores in a name
- Path related:
  10. File path without numbers
  11. How many '/' (Depth of file)
  12. Number of '.' in a path

The list of features includes 2 categorical features namely "File path without numbers" and "File Type" which need to be tokenized and encoded in order to utilize them in the ML models. To do so, we use CountVectorizer, which applies a bag of words approach, within the Scikit Learn library. CountVectorizer first tokenizes the list of values provided (paths or file types) to create a vocabulary that is used to build a vector indicating the occurrence of each term in each sample.

For example, Table 1 shows the sample vocabulary for the following path list ['/Android/Whatsapp/Sent', '/Android/Telegram'].

CountVectorizer also calculates document frequency (df) of each term. In other words, how many documents contain a term. The following two parameters can be used to control the vocabulary size based on document frequency:

- **max_df**: Allows ignoring all terms with a frequency or percentage of occurrence higher than the set max_df. This helps to avoid including a term that occurs in most of the instances.
- **min_df**: Allows ignoring all terms with a frequency or percentage of occurrence lower than the set min_df. This helps to avoid including rare terms in the vocabulary.

For the "File Type" feature, these parameters are not set to ensure all file types are captured especially that the set is limited. However, for the "File Path" feature, these parameters are set based on experimentation during hyperparameter tuning which will be discussed in a dedicated sub-section. Also, to further reduce the occurrence of similar terms, all terms are transformed to lower case and numbers are stripped.

Post vectorization, the actual feature space will increase. For instance, File Type will produce a feature for each of the detected file types, e.g. vect_ext_bmp, vect_ext_jpeg, and a file path of/Andriod/Whatsapp/Sent could produce vect_path_whatsapp vect_path_sent feature. To avoid information leakage, vectorization is applied within a pipeline fed into cross-validation. Thus, the exact number of features will vary depending on the sample and the parameters used. Yet, to get a sense of the number of features, encoding was applied on all the data, and it resulted in 187 features.

Including irrelevant features in the data degrades the predictive performance of a model (Witten et al., 2011). In addition, it increasing fitting and testing time especially with large feature sets. To select the best set of features, Recursive Feature Elimination and

Fig. 3. Sample XRY Log File entry of a video file and a text file.

Table 1
Example of the vectors created from samples S1, S2.

Sample	Android	Whatsapp	Sent	Telegram
S1	1	1	1	0
S2	1	0	0	1

Cross-validation (RFECV) from the Scikit Learn library is used on the validation dataset.

RFECV essentially automates the task of manually testing different sets of features to select the set that produces the best performance. RFECV starts with the complete set of features, then eliminates the weakest feature(s), then repeats until the minimum number of required features is reached.

RFECV ranks all features based on their importance. Features that result in the best test results are all given a rank equal to '1'. Table 2 represents an extract of the RFECV results, and shows that all created features were selected while only dropping 12 out of the 75 vocabularies of the path feature. The dropped vocabularies are not base features, and Fig. 4 shows that including them has minor impact on performance as the graph slightly fluctuates around an F1-score of 0.969 after 23 features is reached while reaching its

Table 2
Feature selection results.

Feature	Ranking	Selected
size_KB	1	TRUE
EXIF	1	TRUE
Xres	1	TRUE
Yres	1	TRUE
day_delta	1	TRUE
name_len	1	TRUE
name_num_per	1	TRUE
name_uscore_per	1	TRUE
path_dep	1	TRUE
path_dots	1	TRUE
vect_ext_bmp	1	TRUE
vect_ext_jpeg	1	TRUE
vect_ext_mp3	1	TRUE
vect_ext_ogg	1	TRUE
vect_ext_png	1	TRUE
vect_ext_webp	1	TRUE
vect_ext_xml	1	TRUE
…	…	…
vect_path_xfv	2	FALSE
vect_path_emoji	3	FALSE
…	…	…

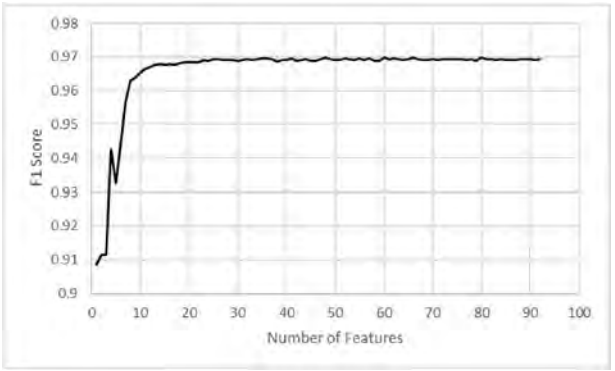

Fig. 4. F1-Score vs. Number of Features.

maximum of 0.9698 at 80 features.

Therefore, the set of 12 features initially created is used in our research.

3.8. Hyper-parameter tuning

Regarding hyper-parameter tuning, hyper-parameters are variables of the classification algorithm and not the designed model, therefore they should be set before training the model. One possible way to set the parameters is to manually try different parameters on a validation set and select the ones that yields the best performance. Alternatively, GridSearchCV from the Scikit Learn library can be used to automate this process (Géron, 2019). GridSearchCV tries all possible combinations of the provided range of parameters with cross validation, and then selects the best set. Table 3 shows the selected hyperparameters.

3.9. Performance metrics

The performance of the tested classification algorithms is compared according to the following metrics: F1-Score (F1), Recall (R), Precision (P) and Accuracy (ACC) (Gómez, 2012; Marturana et al., 2011a; Mohammad and Alq, 2019).

1. Precision (P): is the ratio of true positive (TP) predictions out of all positive predictions (TP and FP (False Positive)).

$$P = \frac{TP}{TP + FP} \tag{1}$$

2. Recall (R): also referred to as sensitivity is the ration of TP predictions out of the actual positive items (FN is False Negative).

$$R = \frac{TP}{TP + FN} \tag{2}$$

3. F1-Score (F1): is a harmonic mean of precision and recall, thus a good F1-Score requires a good score on both of recall and precision simultaneously.

Table 3
Selected hyperparameters for each model.

Model	Hyperparameters
KNN	‘ clf__leaf_size': 10 ‘clf__n_neighbors': 3 ‘clf__p': 1, ‘preprocessor__path__vect__max_df': 0.9 ‘preprocessor__path__vect__min_df': 0.005
CART	‘clf__criterion': ‘entropy' ‘clf__max_depth': 16 ‘clf__min_samples_leaf': 1 ‘clf__min_samples_split': 2 preprocessor__path__vect__max_df': 0.95 ‘preprocessor__path__vect__min_df': 0.005
NB	‘clf__alpha': 1.0 ‘preprocessor__path__vect__max_df': 0.9 ‘preprocessor__path__vect__min_df': 0.005
SVM	‘clf__C': 10 ‘clf__gamma': ‘scale' ‘clf__kernel': ‘rbf' ‘preprocessor__path__vect__max_df': 0.9 ‘preprocessor__path__vect__min_df': 0.005
RF	‘clf__bootstrap': False ‘clf__max_depth': None ‘clf__max_features': ‘auto' ‘clf__min_samples_leaf': 2 ‘clf__min_samples_split': 5 ‘clf__n_estimators': 100 ‘preprocessor__path__vect__max_df': 0.9 ‘preprocessor__path__vect__min_df': 0.005
NN_MLP	‘clf__activation': ‘tanh' ‘clf__alpha': 0.0001 ‘clf__hidden_layer_sizes': (80, 50) ‘clf__learning_rate': ‘constant' ‘clf__learning_rate_init': 0.001 ‘clf__solver': ‘adam' ‘preprocessor__path__vect__max_df': 0.95 ‘preprocessor__path__vect__min_df': 0.005

$$F1 = \frac{2PR}{P+R} \tag{3}$$

4. Accuracy (ACC): is the ratio of correctly classified items out of all items (TN is True Negative).

$$ACC = \frac{TP+TN}{TP+TN+FP+FN} \tag{4}$$

The scoring measurement used in cross validation is F1-Score as it combines two important metrics for our case, namely recall and precision. A good classifier shall be able to recall as much as possible of the files of interest, yet it should still preserve precision to minimize the number of uninteresting files that it falsely marks as interesting.

4. Experiments and results

Applying the approach proposed to answer the research question results in a set of classification models that take file metadata as input and produce a label for the file ("Interesting" or "Not Interesting"). To judge the performance of these models, 10-fold cross validation is used on a set of 1,799,055 records where interesting files have a 6% prevalence. So, 10 experiments are run for each classifier every time training on around 1,619,150 records and testing on 179,905 records. This section describes the results obtained from these experiments by covering the predictive performance of the different models, and the execution times. This section will cover the performance of the classifiers across all the mentioned metrics, and conclude with a brief summary of the performance results.

4.1. F1-score

Fig. 5 shows boxplots of all the F1-Scores obtained during cross validation for each classifier. The figure shows that all classifiers exhibit very minor variations in F1-Scores all with standard deviation less than or equal to 0.0024. NB exhibit the highest variation with a range of 0.007 or 0.8%. RF achieves the highest F1-Score at 0.9861, which is slightly higher than the second ranking classifier, KNN that scored 0.9840. CART follows at 0.9837, then NN_MLP at 0.9744, then SVM at 0.9735. NB score the lowest at 0.8795, thus 12% less than the highest achieved score.

4.2. Recall

Fig. 6 shows that recall results were consistent across trials with standard deviation and a range respectively less than or equal to 0.024 and 0.0043 except for NN_MLP. Although, the later achieves acceptable standard deviation at 0.0098 and a range of 0.031, it still exhibits the most variation when compared to the other classifiers. As can be inferred from the standard deviation and from the graph most of the trials fall close to the mean. NB achieves the highest recall score at 0.9932 although it has achieved the lowest F1-Score, this indicates that it did not perform as well as the other classifiers in precision as will be seen in the next section. In other words, NB was able to identify 99.32% of the files of interest. SVM ranks second at 0.9913, followed by RF at 0.9881, followed by KNN at 0.9851, then CART at 0.9835. NN_MLP comes in last at 0.9827, so it identified around 1% less files of interest than the best performer.

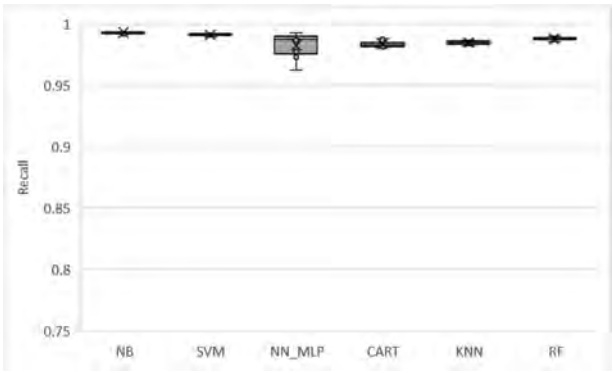

Fig. 6. Model comparison - recall box plots.

4.3. Precision

Fig. 7 shows that SVM, CART, KNN, and RF exhibit very low variation in results with a standard deviation and range less than or equal to 0.0020 and 0.0060 respectively. NB exhibits slightly more variation with a standard deviation 0.0037 and a range of 0.011. NN_ML exhibits behavior similar to Recall. It is worth noting that the interquartile range in precision is closer to the minimum while in Recall it is closer to the maximum, thus when the F1-Score, which is a harmonic mean of these two metrics, exhibits very low variation as seen previously. RF scores the highest mean precision at 0.9841, followed closely by CART and KNN at 0.9838 and 0.9830 respectively. NN_MLP comes next at 0.9664, followed by SVM at 0.9562. NB performs significantly worse than the other classifiers scoring 0.7891, thus in comparison RF exhibits around a 25% improvement on NB.

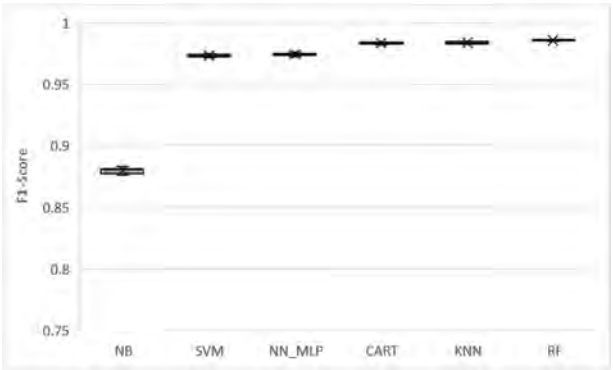

Fig. 5. Model comparison - F1 scores box plots.

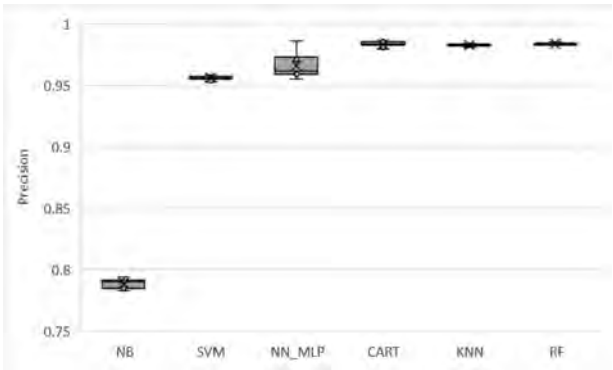

Fig. 7. Model comparison - precision box plots.

4.4. Accuracy

Fig. 8 shows that all models exhibit very low variation in accuracy scores with the highest standard deviation exhibited by NB at 0.0004. RF performs the best at 0.9983, followed closely by KNN and CART that achieve an accuracy of 0.9981. NN_MLP and SVM achieve slightly lower accuracy scores at 0.9969 and 0.9968. NB comes in last yet with a high accuracy score of 0.9838. A key reminder here is that the dataset is unbalanced with only 6% prevalence for the interesting class. This causes accuracy to be a misleading performance measure on its own for such cases. To clarify that, consider that a 7th dummy classifier is added to the set, this classifier always labels files as Not Interesting. The resulting accuracy score of this dummy classifier would be 0.94 or 94% as it would have classified the majority of the files correctly.

4.5. Summary

Since the objective is to maximize both Recall and Precision so that the examiner is presented with maximum files of interest and the minimum files that are not interesting, the F1-Score is used to rank the predictive performance of the algorithms for this research question. Accordingly, RF ranks 1st with an F1-Score at 0.9861, Precision at 0.9841, Recall at 0.9881 and Accuracy at 0.9983. KNN ranks 2nd, CART 3rd, NN_MLP 4th, SVM 5th and NB last. All classifiers except NB exhibit consistently very good classification and consistent performance with an F1-Score equal to or larger than 0.9735.

Table 4 summarizes the mean scores achieved by all classifiers against the performance metrics, and Table 5 summarizes the standard deviation exhibited by each classifier against each metric.

4.6. Execution time

Aside from predictive performance, execution time is worth considering as large execution times can hinder model development efforts or cause delays in deployed systems. Fig. 9 displays the mean total execution time per cross validation split. The figure shows that KNN is the slowest with a total execution time of 697.6 min or 11.63 h. SVM is the second slowest with an execution time of 611.4 min or 10.19 h. Then execution time significantly drops where RF requires 19.7 min, followed by NN_MLP at 7 min, then CART at 1.4 min, and finally NB at 0.9 min.

5. Evaluation and discussion

The performance of several models was tested and the results reported in the previous section. The obtained results achieve the objectives set by this work and support the research hypothesis, namely: (i) File metadata can indicate if a file extracted from a smart phone should be examined; (ii) ML can be used to design triage tool for files extracted from smart phones; (iii) Different ML classification algorithms will result in different predictive performance.

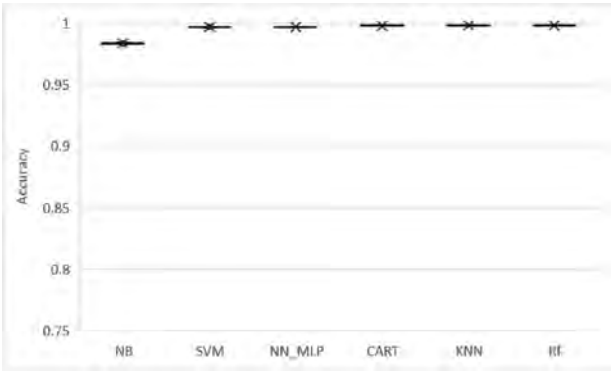

Fig. 8. Model comparison - accuracy box plots.

Table 4
Model comparison — summary of performance metrics.

Classifier	F1-Score	Precision	Recall	Accuracy
NB	0.8795	0.7891	**0.9932**	0.9838
SVM	0.9735	0.9562	0.9913	0.9968
NN_MLP	0.9744	0.9664	0.9827	0.9969
CART	0.9837	0.9838	0.9835	0.9981
KNN	0.9840	0.9830	0.9851	0.9981
RF	**0.9861**	**0.9841**	0.9881	**0.9983**

Table 5
Model comparison - summary of standard deviation.

Classifier	F1-Score	Precision	Recall	Accuracy
NB	0.0024	0.0037	0.0009	0.0004
SVM	0.0011	0.0017	0.0008	0.0001
NN_MLP	0.0011	0.0095	0.0098	0.0001
CART	0.0006	0.0020	0.0024	0.0001
KNN	0.0007	0.0010	0.0014	0.0001
RF	0.0006	0.0008	0.0012	0.0001

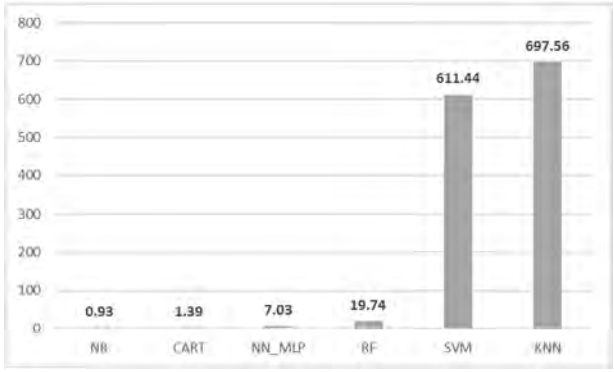

Fig. 9. Model comparison - mean fit and score time (minutes).

5.1. Discussion

Six different classification models were tested and common performance metrics were measured to verify the research hypothesis.

Evaluation results show that RF performs the best with its ability to on average recall 98.81% of the files of interest with a precision of 98.41%. Thus, resulting in an average F1-Score of 0.9861 which represents a 0.21% improvement on the second-best performing classifier. RF performed consistently across the 10 cross validation cycles. For a triage functionality this is sufficient to help an examiner quickly process a mobile phone, a more thorough examination can be conducted at a later stage to ensure completeness.

KNN comes in the second with an F1-Score of 0.9840, a precision of 98.30%, and recall of 98.51%. So, it recalled 0.3% less files than RF, and labelled 0.11% more files than RF as interesting while they are not. KNN also performed 35 times slower than RF.

CART, NN_MLP and SVM also perform well with the least performing (SVM) achieving an F1 score of 0.9735, precision of 95.62%

and recall of 99.13%. Compared to RF, RF achieves a 1.3% enhancement on F1 score, incorrectly labels 2.8% less files as interesting, however it identifies 0.3% less interesting files than SVM. SVM also performs 31 times slower than RF.

NB performs the worst with an F1-Score of 0.8795, precision of 78.91%, and recall of 99.32%. Although NB achieves the highest recall, it also scores significantly lower on precision. In practice, this means that although NB will detect more interesting files, it will also bombard the examiner with significantly more uninteresting files.

The obtained results show that 5 out of the 6 tested algorithms consistently perform well in classifying files extracted from smart phones. Therefore, it can be concluded that ML can be used on file metadata to classify files extracted from a smart phone.

5.2. Applicability of results

To deem results applicable, (1) the used data has to be similar to real case data, and (2) results have to be generalizable. To satisfy the first condition, metadata of files obtained from devices involved in real criminal cases is used to generate the 1,998,950 records used in this research. To satisfy the second condition, 10-fold cross validation with stratified sampling is used. The results for each fold are reported, and the variation of results is considered when evaluating performance. As detailed in Section 4, the models performed consistently across the cross-validation folds. Therefore, the proposed methodology can be deemed applicable in real cases.

6. Conclusion and future work

A triage tool that helps examiners quickly identify files of interest on smart phones aids in overcoming challenges imposed by increasing data sizes and limited investigation time. Such a tool can help in resolving time critical cases and lab backlog.

Surveyed literature showed that several approaches based on classical automation are proposed to design such a tool. However, these techniques lack flexibility in adapting to changes in files and files systems. More flexible techniques based on ML were proposed to triage digital devices. Only one recent effort proposed a similar yet more basic approach to this paper and based its results on files generated and extracted from a computer.

This research effort proposed the use of ML to classify files extracted from smart phones based on their metadata. This paper detailed the proposed approach, and presented the results obtained from the experiments conducted using 6 different ML classification algorithms.

Data used in this paper was forensically extracted from smart phones linked to real terrorism cases. Then file metadata was extracted and labels were added to generate the raw dataset used in this research. Then feature engineering and selection was applied, followed by model hyper-parameter tuning. Finally, cross validation was used to train and test the performance of the different classification models.

Results showed that 5 out of the 6 ML models performed well in classifying files extracted from mobile phones. Thus, the obtained results supported the hypothesis of this paper. This research effort presents a robust approach to triage files extracted from a smart phone, which can also be considered a proof of concept to develop ML based triage tools for files extracted from smart phones.

Another important contribution of this papers is the presented approach for systematic feature selection.

Since only Android OS devices were considered in the experimentation, future research efforts can test this approach on other smart phone operating systems such as iOS as well as on other smartphone's datasets. Also, considering deep learning classification algorithms is an interesting area that can be explored in future research efforts. Another further area of research can be exploring the possibility and effect of augmenting metadata features with features extracted from file content to enhance predictive performance.

References

Bertè, R., Marturana, F., Me, G., Tacconi, S., 2012. Data Mining Based Crime-dependent Triage in Digital Forensics Analysis. https://doi.org/10.13140/2.1.3119.9680.

Dalins, J., Tyshetskiy, Y., Wilson, C., Carman, M., Boudry, D., 2018. Laying foundations for effective machine learning in law enforcement Majura – a labelling schema for child exploitation materials. Digit. Invest. 26 https://doi.org/10.1016/j.diin.2018.05.004.

Deloitte, 2017. Global Mobile Consumer Trends, second ed. 2017. https://www2.deloitte.com/content/dam/Deloitte/us/Documents/technology-media-telecommunications/us-global-mobile-consumer-survey-second-edition.pdf. (Accessed December 2020).

Du, X., Scanlon, M., 2019. Methodology for the automated metadata-based classification of incriminating digital forensic artefacts. Proceedings of the 14th International Conference on Availability, Reliability and Security 1–8. https://doi.org/10.1145/3339252.3340517. August 2019 Article No.: 43.

Faheem, M., Le-Khac, N.-A., Kechadi, M.-T., 2014. Smartphone forensics analysis: a case study for obtaining root access of an android SamSung S3 device and analyse the image without an expensive commercial tool. J. Inf. Secur. 5 (3), 83–90. https://doi.org/10.4236/jis.2014.5300.

Garfinkel, S., 2013. Digital media triage with bulk data analysis and bulk_extractor. Comput. Secur. 32, 56–72. https://doi.org/10.1016/j.cose.2012.09.011.

Garfinkel, S., Parker-Wood, A., Huynh, D., Migletz, J., 2011. An automated solution to the multiuser carved data ascription problem. Inf. Forensics Secur. IEEE Trans. 5, 868–882. https://doi.org/10.1109/TIFS.2010.2060484.

Géron, A., 2019. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, second ed. O'Reilly Media, Inc. 2019, ISBN 9781492032649.

Gómez, L., 2012. Triage In-Lab: Case Backlog Reduction with Forensic Digital Profiling. https://www.researchgate.net/publication/265748911_Triage_in-Lab_case_backlog_reduction_with_forensic_digital_profiling. (Accessed December 2020).

Hitchcock, B., Le-Khac, N.-A., Scanlon, M., 2016. Tiered forensic methodology model for Digital Field Triage by non-digital evidence specialists. Digit. Invest. 16 (Suppl. ment), S75–S85. https://doi.org/10.1016/j.diin.2016.01.010, 29 March 2016.

Khan, M., Wakeman, I., 2006. Machine learning for post-event timeline reconstruction. https://www.researchgate.net/publication/228941446_Machine_Learning_for_Post-Event_Timeline_Reconstruction. (Accessed December 2020).

Lim, S., 2019. Average Storage Capacity in Smartphones to Cross 80GB by End-2019 [WWW Document]. Counterpoint. https://www.counterpointresearch.com/average-storage-capacity-smartphones-cross-80gb-end-2019/.

Marturana, F., Bertè, R., Me, G., Tacconi, S., 2011a. A quantitative approach to triaging in mobile forensics. In: IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications 16-18 Nov. 2011, Changsha, China. https://doi.org/10.1109/TrustCom.2011.75.

Marturana, F., Bertè, R., Tacconi, S., Me, G., 2011b. Mobile Forensics "triaging": new directions for methodology Authors. In: Proceedings of VIII Conference of the Italian Chapter of the Association for Information Systems (ITAIS). Academic Press. https://doi.org/10.13140/2.1.1760.0968, 2011.

Milosevic, N., Dehghantanha, A., Choo, K.-K.R., 2017. Machine learning aided Android malware classification. Comput. Electr. Eng. 61 https://doi.org/10.1016/j.compeleceng.2017.02.013.

Mohammad, R., 2019. An enhanced multiclass support vector machine model and its application to classifying file systems affected by a digital crime. J. King Saud Univ. 1, 12. https://doi.org/10.1016/j.jksuci.2019.10.010.

Mohammad, R., Alq, M., 2019. A comparison of machine learning techniques for file system forensics analysis. J. Inf. Secur. Appl. 46, 53–56. https://doi.org/10.1016/j.jisa.2019.02.009.

Msab Products [WWW Document], n.d. https://www.msab.com/products/xry/.

StatCounter, 2020. Mobile Operating System Market Share Worldwide [WWW Document]. https://gs.statcounter.com/os-market-share/mobile/worldwide.

Sester, J., Hayes, D., Scanlon, M., Le-Khac, N.-A., 2021. A comparative study of support vector machine and neural networks for file type identification using n-gram analysis. Forensic Sci. Int.: Digit. Invest. 36 (Suppl. ment), 301121. https://doi.org/10.1016/j.fsidi.2021.301121. April 2021.

Witten, I.H., Frank, E., Hall, M.A., 2011. Data Mining: Practical Machine Learning Tools and Techniques.

Witteman, R., Meijer, A., Kechadi, M.-T., Le-Khac, N.-A., 2016. Toward a New Tool to Extract the Evidence from a Memory Card of Mobile Phones, 2016 4th International Symposium on Digital Forensic and Security (ISDFS), 25-27 April 2016. Little Rock, AR, USA. https://doi.org/10.1109/ISDFS.2016.7473533.