Privacy concerns in IoT systems with the use of AI techniques

Fabio Palomba
fpalomba@unisa.it
Universitá degli studi di Salerno
Salerno, Italy

Giammaria Giordano
ggiordano@unisa.it
Universitá degli studi di Salerno
Salerno, Italy

Biagio Boi
Gigi Jr Del Monaco
b.boi@studenti.unisa.it
g.delmonaco1@studenti.unisa.it
Universitá degli studi di Salerno
Salerno, Italy

1 ABSTRACT

The work tries to discover problems related to privacy in IoT systems; in particular those which make use of smart assistances like Alexa or Google Home. The project studies all the existing works in order to understand if these systems have a good level of reliability and integration between each other. A complete refactoring of a system has been proposed; in particular by thinking about the MLOps paradigm, which guarantee a good degree of control.

2 INTRODUCTION

The evolution of smart devices over last years has increased exponentially and the introduction of these devices in the house is progressively growing. The major problem related to these devices is that usually the privacy is not considered, although there are a lot of regulations (just see the GDPR) that describe how the user data have to be stored and who can access to these data. Starting from these two points we've decided to understand what happens within the context of smart assistance. Various projects have been developed and all these projects try to discover a correlation between packets and possible patterns to discover the conversations and the presence of someone inside the home; which can be seen as serious privacy violation.

3 GOAL OF THE PROJECT

The goal of the project is to assess the reliability of developed projects; in particular, an initial possible integration has been evaluated and consequentially; since this integration has not been possible, a comparison between dataset and pipeline automation has been proposed.

4 METHODOLOGICAL STEPS CONDUCTED TO ADDRESS THE GOALS

In order to implement the tool, we've decided to follow each of these steps:

(1) Consider the current state of art in order to retrieve useful informations. This step is important to produce knowledge to better perform the next steps;
(2) Analyze the existing datasets to achieve feature engineering;
(3) Apply normalization techniques (data cleaning, data balancing);
(4) Implementation and training of a ML model by considering different approaches to find the best fit model for our problem;
(5) Analyze, monitor and compare the results of each model by using ML Flow tool.

Clearly, all these steps will be conducted using a MLOps approach.

4.1 First comparison between projects

As introduced, the project started with the comparison among already developed projects and datasets in order to extract important feature from each project. The first considered project has been that one developed by Kennedy et al. [1], which examine a passive attack on home smart speaker able to infer users' voice commands. The project focuses on a particular metric, the so called semantic distance.Even if this project has a focus on a critical point of privacy, it seems quite complicated find out the reasoning inside the semantic distance; that's why we have decided to not extend this project. The second project considered is "Alexa real time analyzer", developed by Boi B. which give us a better vision on caught data from network and considered metrics; the model created from this project seems to be powerful on data retrieved by the creator; for this reason we want to assess if the model works good also on other data.

4.2 Dataset Creation

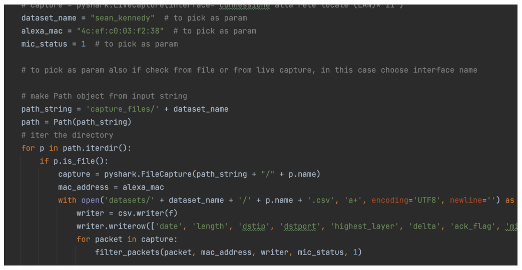
Since the Kennedy project offers a good set of captured packets we've decided to use these files to produce a dataset which is compatible with the format requested from the Boi's project. In particular, since the existing script was able to perform live capturing from a live scenario, we have modified this script to guarantee both capture: from files and from live context; in this way we have automated the creation of dataset starting from both contexts. The decision of creating a dataset from the captured files has been taken for three main reasons:

- To check if the model created from the second project works well on unseen data;
- To create a new model (using various ML technique, that we will see below) based on these new data;
- To assess the possibility of automatically collect new data.

In this phase no check mechanisms has been take in place; in the following subsection we will see these mechanisms.

4.3 Data Debt & Solution

As introduced in the previous subsection, the data are pushed into a dataset by transforming well-structured data, such as Wireshark packet, into a row of a csv file. In particular, by following the



```
# capture = pyshark.LiveCapture(interface='Connessione alla rete locale (LAN)* 11')
dataset_name = "sean_kennedy"  # to pick as param
alexa_mac = "4c:ef:c0:03:f2:38"  # to pick as param
mic_status = 1  # to pick as param

# to pick as param also if check from file or from live capture, in this case choose interface name

# make Path object from input string
path_string = 'capture_files/' + dataset_name
path = Path(path_string)
# iter the directory
for p in path.iterdir():
    if p.is_file():
        capture = pyshark.FileCapture(path_string + "/" + p.name)
        mac_address = alexa_mac
        with open('datasets/' + dataset_name + '/' + p.name + '.csv', 'a+', encoding='UTF8', newline='') as
            writer = csv.writer(f)
            writer.writerow(['date', 'length', 'dstip', 'dstport', 'highest_layer', 'delta', 'ack_flag', 'mi
            for packet in capture:
                filter_packets(packet, mac_address, writer, mic_status, 1)
```

Figure 1: Parser able to do live or files capture

documentation related to the Boi's project, the dataset has the following feature:

- **date**: the date in which the packet has been collected, in the format YYYY-MM-DD hh:mm:ss:ms;
- **length**: the length of the packet, it includes just the payload (in case of ack packest it's equal to zero);
- **dstip**: the destination ip, collected to analyze the owner of the server to which the packet is directed;
- **dstport**: the destination port;
- **highest_layer**: the protocol used, in order to parse the protocol into an integer we will use the following mapping:
  - 0 - SSL
  - 1 - TCP
  - 2 - DATA
  - 3 - HTTP

  Notice that all the packets that use other protocol are discarded since they have no meaning for our purpose;
- **delta**: the time occured from the previous packet of the same stream;
- **ack_flag**: the acknowledge flag; it is equal to 1 if the packet contains an ack;
- **microphone**: the status of the microphone, it is equal to 1 if the microphone is active, 0 otherwise;
- **content_type**: the type of content sent, it is valorized only if the packet is sent over SSL;
- **synchronized**: status of synchronization of the device, it is equal to 1 if Echo has been previously associated to an account, 0 otherwise.

Different problems have been issued here; in order to guarantee a well-structured dataset, that avoid data debt, we have decided to do some checks on dataset created from capture files. Following problems have been faced up and solved by applying different techniques:

- When the script tried to analyze packets from the Haipeng and Sean dataset, it was unable to classify application data packets since the highest layer used from the version of Echo Dot was TLS instead of SSL. For this reason, a modification to the parser has been applied in order to correctly classify these packets. Furthermore, other modifications have been introduced to notificate the Data Engineer that the parser wasn't able to assign a class to the packet.
- During the dataset overview various None values have been retrieved on the class column (which is the target class) and on content type feature; we've decided to delete the rows which contains None values on target class and notificate the Data Engineer in order to check the behavior of the parser, and consequentially modify it in order to optimize the classification; while we have decided to put 0 to content type since the packets that have null content type are related to synchronization and acknowledge packets. We avoided the possibility to use Data Imputation techniques to substitute the value of this target class column, since it is the most important column of the dataset. Anyway, a final check has been introduced if some new technical modification happens to Echo dot communication system.

4.4 Feature Engineering

After solved problem on structure, quality and integrity of data captured; we will now discuss about the feature engineering process achieved. As first step we checked if data contains feature that are not relevant for the aim of our project - in particular, the date of packet capture and the destination ip seems to be strictly related to the particular instance of the packet; for this reason we removed these feature. Furthermore, after conducting an analysis that considers the feature distribution, we've decided to remove the synchronization feature since there is no packet captured without synchronization. This clear behavior of *low variance removal* is caused by the Echo Dot architecture, which does not admit any data transferring without Amazon account association. This last consideration, let us think that this feature might be directly removed from the feature captured at parsing time since it is totally not relevant. Once we removed the not relevant feature, we need to balance the dataset; since we're working with three different structure dataset they are really diverse among them.

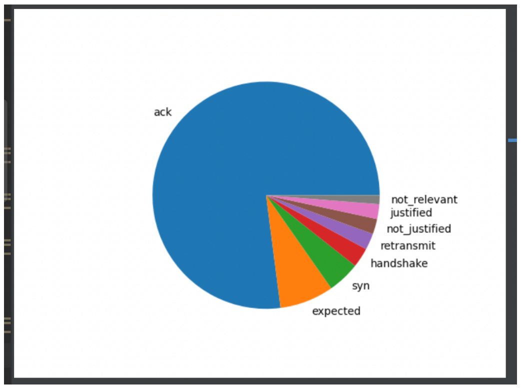


Figure 2: Distribution of Biagio Boi dataset

A direct application of SMOTE technique is useless at this time since there are too much diversity in data distribution. For this reason do some considerations on these particular instances, that might be true in general:

- Since we have a lot of ack packets in all three datasets we apply some reduction technique to truncate these data (basically an ack packet is just a confirmation of the data sent from the Echo Dot, which is not relevant for our purpose) by applying 80% deletion.

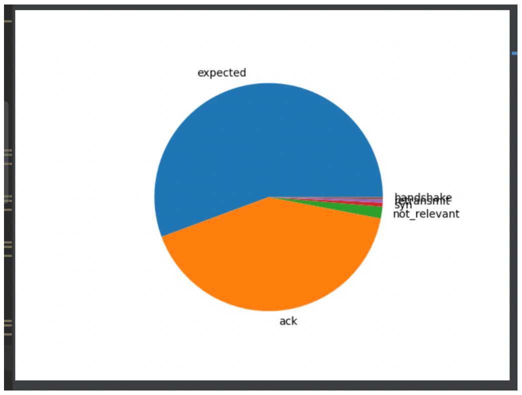


Figure 3: Distribution of Haipeng Li dataset

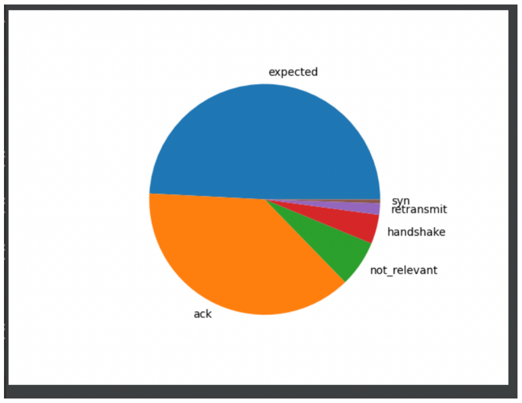


Figure 4: Distribution of Sean Kennedy dataset

- Another class which is too many present is the expected one; in Biagio Boi dataset this is not true since he seems to collect a good variety of packets among the other classes, specifically for the purpose. The other two datasets, instead, since have collected data from a normal conversation context have a lot of these packets. We can reasonably thinking that this is the most likely frequent dataset during collection phase, so we truncate 85% of these packets resulting from the union of all three datasets.

The following pie chart shows the new distribution of the conjunction of all three dataset after applying deletion techniques.

Now, after deleting the extra data, which are not good for our purpose, we can now apply SMOTE technique to better balance the final dataset.

4.5 MLFlow

After solved the problem with dataset, we started experiment using MLFlow framework. We started using the tools by creating an experiment for each dataset in order to understand how the models fit the data. In particular, in order to assess the quality of our dataset and models we applied a k-cross fold validation. After some experimentation, we discovered that k=10 is the best value

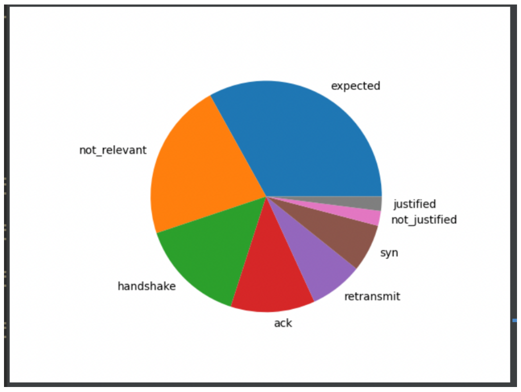


Figure 5: Distribution of final dataset

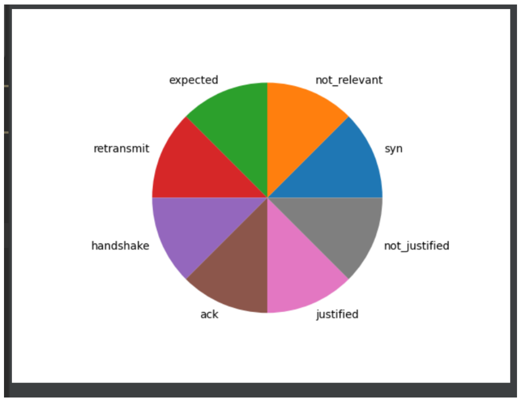
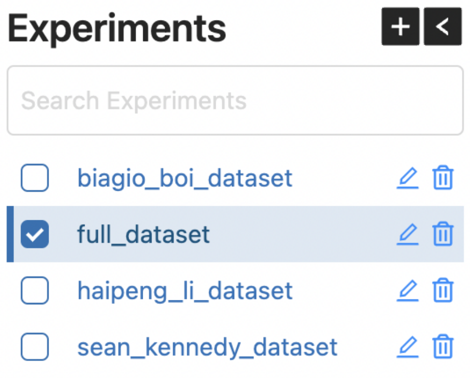


Figure 6: Distribution of final dataset after applying SMOTE


Experiments

Search Experiments

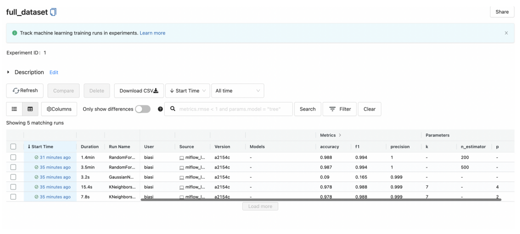
biagio_boi_dataset
full_dataset
haipeng_li_dataset
sean_kennedy_dataset

Figure 7: Created Experiments on MLFlow

for the number of folds to generate. Then, different classifiers have been applied:

- **KNeighborsClassifier** - with number of neighbors equals to 7, since we have 7 classes to distinguish and variable Minkowski metric parameters (2 and 4)
- **RandomForestClassifier** - with number of estimators variable (200 or 500)

- **GaussianNB**



full_dataset

Experiment ID: 1

Showing 5 matching runs

							Metrics			Parameters		
Start Time	Duration	Run Name	User	Source	Version	Models	accuracy	f1	precision	k	n_estimator	p
31 minutes ago	1.4min	RandomFor...	biasi	mlflow_l...	a2154c	-	0.988	0.994	1	-	200	-
35 minutes ago	3.5min	RandomFor...	biasi	mlflow_l...	a2154c	-	0.987	0.994	1	-	500	-
35 minutes ago	3.2s	GaussianN...	biasi	mlflow_l...	a2154c	-	0.09	0.165	0.999	-	-	-
35 minutes ago	15.4s	KNeighbors...	biasi	mlflow_l...	a2154c	-	0.978	0.988	0.999	7	-	4
35 minutes ago	7.8s	KNeighbors...	biasi	mlflow_l...	a2154c	-	0.978	0.988	0.999	7	-	2

Figure 8: Full Dataset Experiment on MLFlow

The results obtained on full dataset seems to be satisfying, especially for the Random Forest Algorithm. Regarding the single dataset, the behavior seems to be the same; which means that the adopted parse and the computer algorithm performs well also on different context, such as those of the Haipeng and Kennedy project. Going deeper, we can analyze the evolution of each metric shown in the table. As we can see, also by applying k-cross validation there is a clear distinction between data, since the accuracy value stays always high independently from the piece of dataset used. It's clear how

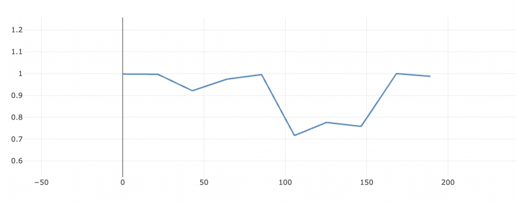


Figure 9: Accuracy evolution on full dataset

other metrics can be used here to assess the quality of produced models, but for didactic purpose we limite our study to this.

5 CONCLUSIONS

The aim of the project has been partially achieved, in particular we've developed a kind of ML Pipeline that follows the MLOps paradigm, where we distinguish the training phase from the Inference phase. In particular, the inference phase hasn't been covered by this project. Some study on MLFlow have been conducted, in particular we've tried to use the Model function of this framework to store the created models without any success due to problems related to database issues. Further conclusions regards the quality of data used; in particular the parser used for the transformation of packets into well-structured dataset is projected by starting from assumption on the structure of packets. These assumption are not well documented, by causing the un-reliability of the entire models built upon this parser. A complete tool able to block the packets cannot be yet released, but a strong pipeline for capturing packets and transforming these packets in well-structured data and then cleaning and preprocessing techniques have been proposed.

REFERENCES

[1] Sean Kennedy, Haipeng Li, Chenggang Wang, Hao Liu, Boyang Wang, and Wenhai Sun. I can hear your alexa: Voice command fingerprinting on smart home speakers. In *2019 IEEE Conference on Communications and Network Security (CNS)*, pages 232–240, 2019.