Name: Alex González Student ID: 969926

COMP90073 Security Analytics Semester 2, 2020

Project 2: Machine learning based cyberattack detection

In this project are explored two machine learning topics regarding cyberattacks. In the work corresponding to Task 1 are analysed different combinations of features and machine learning models for anomaly detection. The anomalies in this case correspond to traffic originated by botnets. In this task is utilised an **unsupervised** machine learning approach. On the other hand, in the Task 2 an adversarial sample is generated for evading the detection of a **supervised** machine learning model.

Task 1

1 Data overview

The data in this project consists in network traffic captured under the NetFlow[1] format. In the datasets provided are present both normal traffic and traffic originated by botnets. As it was mentioned above, this task corresponds to an unsupervised approach where no labels are provided for the data. The *Table 1* shows a general description of the dataset fields.

N°	Name	Description	Type
1	Timestamp	Timestamp of the traffic.	Categorical
2	Duration	Duration in seconds.	Numerical
3	Protocol	Internet Protocol (i.e.: ARP, IPv6, etc.)	Categorical
4	SrcIPAddress	Source IP address.	Categorical
5	SrcPort	Source port.	Categorical
6	Direction	Direction of the traffic.	Categorical
7	DestIPAddress	Destination IP address	Categorical
8	DestPort	Destination port.	Categorical
9	State	State of the traffic.	Categorical
10	SrcTypeService	Source type of service.	Categorical
11	DestTypeService	Destination type of service	Categorical
12	TotalPackets	The number of total packets.	Numerical
13	TotalBytes	Bytes transferred in both directions.	Numerical
14	SourceBytes	Bytes transferred from source to destination.	Numerical

[1] NetFlow is a feature of Cisco routers that provides the ability to collect IP network traffic.

Table 1 Dataset overview

Some fields are specific in the context of network traffic and it is not needed a deeper comprehension give the project scope. For Task 1 they were provided three datasets for the purposes of training, test and validation. The size of these datasets is shown in *Table 2*.

Dataset	N# of rows
Training	13.882.035
Test	1.053.845
Validation	940.062

Table 2 Size of datasets

2 Feature generation and selection

The file *A1_preprocessing-1* contains the following pre-processing subtasks:

- It was cleaned and completed the field *Duration*: For example, this field contained values with spaces such as ' <-' or '   ?>'. Also, it was determined the direction for rows with '<?>' based in the information of other rows.
- They were added the following features: Packets by Second, Total Bytes by Second and Source Bytes by Second (Second is represented by the Duration feature).

The file *A1_preprocessing-2* contains the following pre-processing subtasks:

- Numerical features: They were used the 4 numerical features of the dataset. These features were scaled using a the StandardScaler method available in Scikit Learn. This method normalizes the data with a mean equal to 0 and standard deviation equal to 1.
- Categorical features: Several trade-offs were taken in consideration to pre-processing these types of features. The most correct approach should be One Hot Encoder. Even though, the dataset is large, and it is important to keep manageable for standard computational resources. For example, the feature SrcPort has 97.555 different values in the train dataset. With One Hot Encoder will be generated that quantity of new features. As alternative, it was applied the method LabelEncoder from Skit Learn. The application of this method is not completely correct from a statistical point of view. Under this approach, the encoded categorical features are transformed into numerical features and a correlation is introduced in the model. Even so, it was preferred this method to include all the categorical features in a feasible way.

Regarding Project 1, in that project the dataset consisted in a slightly detailed level of information: networks packets. Instead, NetFlow traffic have a more aggregated level of information based on "conversations" between machines. Also, the approach in Project 1 was focused in detect packets of known protocols casually related to cyberattacks, such as IRC[2] or SMB[3]. Therefore, this information seems to be relevant along with the port utilised. As a new Project 2 finding, it seems to be more significant to work with these aggregated traffic "conversations" to generate information as the duration of the communication and the total bytes transferred.

Continuing with the description of data pre-processing, finally the feature Duration was discarded of the dataset. With this feature, the models tend to detect as anomalies network traffic with high values for this field. Inspecting the validation dataset, it turns obvious that is not the case. Also, the Timestamp feature also was discarded given that it is not expected a correlation useful to predict anomalies. In summary, so far there are available 15 features. In Sections 2.1 and 2.2 are described the two techniques utilised for feature selection.

2.1 PCA[4]

The file *A1_feature_selection_PCA* contains the reduction operation from 15 features to 3 principal components. This latter number was determined with the proportion of variance explained [1]. As it is shown in *Figure 1* with 7 components it is explained the 99% of the original dataset variance. Additionally, it was utilised the method IncrementalPCA (rather than PCA) from Skit Learn due to its convenience to work with large datasets.

```
IncrementalPCA(n_components=10)

print(pca.explained_variance_ratio_.cumsum())

[0.98078331 0.99398511 0.99981839 0.99999984 1.         1.
 1.         1.         1.         1.        ]
```

Figure 1 PCA component selection

2.2 Variance Threshold (VT)

The file *A1_feature_selection_Variance_Threshold* contains the application of Variance Threshold. This technique eliminates the features with low variance assuming that they should not be useful for prediction. In this case, there is not a

[2] IRC (Internet Relay Chat): It is an old application protocol widely used for chat communications.
[3] SMB (Server Message Block): It is a communication protocol for shared access to resources.
[4] PCA: Principal Component Analysis

specific rule to define the threshold. For example, the default value of the Skit Learn implementation is 0. In practice this value should be chosen by inspection. For this case, it was selected as threshold a value of 0.9 which results in the selection of 13 features (out of 15). The discarded features in this case were Direction and SrcTypeService.

3 Models for anomaly detection

Two machine learning techniques used for anomaly detection are iForest [2] and K-Means [3] clustering.

3.1 iForest[5]

For this model, it was utilised the default parameters for the method IsolationForest in Skit Learn. Two of the most relevant parameters are:

- n_estimators: The number of base estimators in the ensemble (default = 100).
- contamination: The proportion of outliers assumed in the training set.

The number of anomalies detected could be especially sensible to this latter parameter. Their set up depends on the knowledge of this specific context. It was tried different values. In section 4.1 are explained further details.

3.2 K-means

For this model the main parameter is the number of clusters. Ideally this parameter should be determined with method such as Elbow or Silhouette Coefficient. The Probably due to the size of dataset. The *Figure 2* shows the first method applied to the train dataset. The graph suggest that it should be chosen 2 or 3 as values for k. Unfortunately, as it is explained in section 4.2, the K-means was not useful for anomaly detection using the most depurate dataset version. A previous version of the dataset with less features should be used with K-means model.

[5] iForest: Isolation Forest.

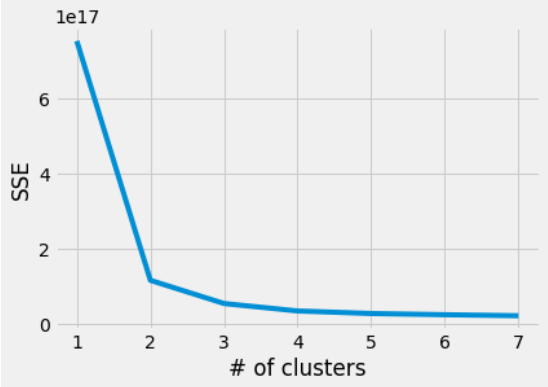

Figure 2 Elbow method

4 Evaluation and results

In *Table 3* is presented a resume of the combination of the different feature selection methods and machine learning models. There are important differences in the model outputs but most importantly, no model has metrics similar to the validation dataset. The model that will be chosen is iForest with contamination parameter equal to 0.015 (in bold in the table). In next sections will be reviewed further details.

Model	# Anomalies	# Source IP	Avg. Duration	Avg. Packets by Sec	Avg. Source Bytes by Sec
iForest (0.01) with PCA	2,074	18	842	7,206	168,781
iForest (0.02) with PCA	21,858	24	941	4,414	103,321
iForest (0.01) with VT	7,329	124	808	6,280	236,690
iForest (0.15) with VT	17,923	133	792	4,638	157,129
iForest (0.02) with VT	23,396	149	836	3,224	110,924
K-Means with PCA	5.708	182	2,780	8	5,269
K-Means with VT	5.710	182	2,780	8	5,268
Training dataset: it is expected that mainly it contains normal traffic.			165	7,318	219,097
Validation dataset: it mainly contains botnet traffic.			*88*	*759*	*20,870*

Table 3 Model results

4.1 iForest

For iForest were tested different values contamination between 0.01 and 0.1. As it is possible to observe in *Table 3* this parameter generated relevant differences even with the same feature selection approach. The *Figure 3* shows the anomaly detection carried out with iForest model (contamination=0.01) under PCA feature selection. It seems that the model is separating the data properly given the available dimensions.

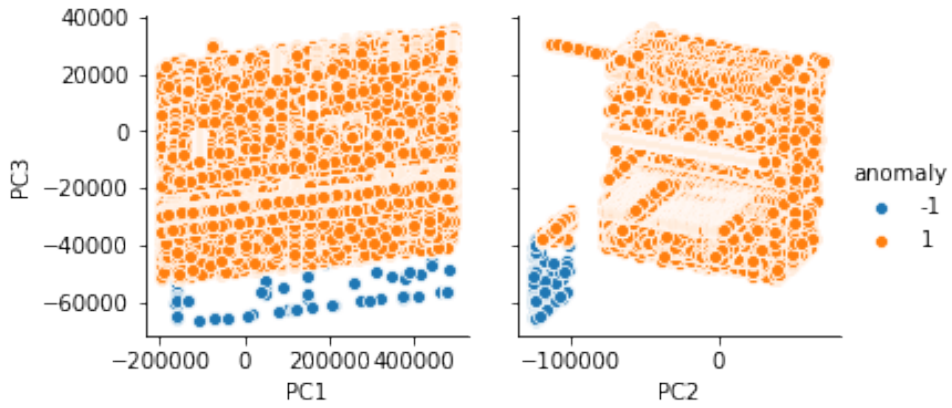

Figure 3 Anomalies - iForest (0.01) with PCA

These models are available in the files: *A1_model_iForest0.01_with_PCA*, *A1_model_iForest0.02_with_PCA*, *A1_model_iForest0.01_with_VT*, *A1_model_iForest0.015_with_VT* and *A1_model_iForest0.02_with_VT*.

Regarding the rank of the anomalies, it was used the method decision function provided by Skit Learn. This function generates a score for each row, while more negative it is more likely to be an outlier. This score was included in all the csv files with the outputs of these models: *A1_test_iForest0.01_PCAorig_feat*, *A1_test_iForest0.02_PCA_orig_feat*, *A1_test_iForest0.01_VT_orig_feat*, *A1_test_iForest0.015_VT_orig_feat* and *A1_test_iForest0.02_VT_orig_feat*. Unfortunately, it was not possible to establish a formal relation between this score and the metrics reviewed in the context of this subject.

4.2 K-means

As it was mentioned above, the most polished dataset does not work properly with K-means. Considering 3 clusters, the models was generating clusters with similar number of elements, therefore it was difficult to speak of anomalies in those scenarios. This happened both for PCA and Variance Threshold feature

selection. As alternative, it was utilised a previous dataset that it was created with a different strategy. In this previous dataset was utilised One Hot Encoder instead of Label Encoder, but just with some selected features. By the way, the results related to K-means was generated with this source (with the exception of *Figure 2*). The *Figure 4* presents the clusters generated for this model using Variance Threshold. Given the data available for the model, it seems that the clustering is working in a reasonable way.

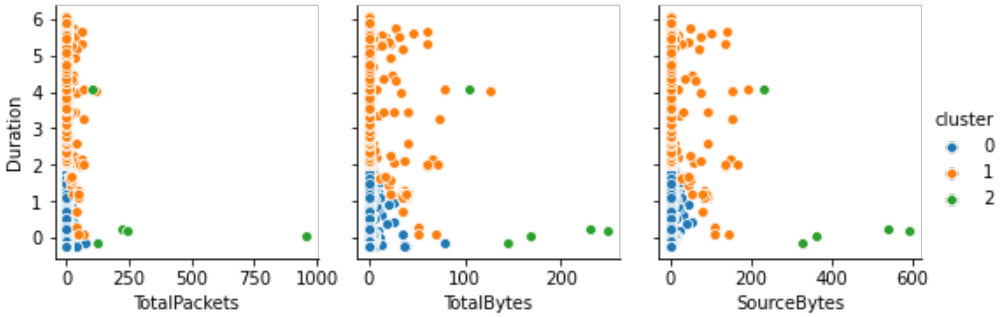

Figure 4 Clusters - k-means with VT

These models are available in the files: *A1_model_Kmean_with_PCA* and *A1_model_Kmean_with_VT*.

Regarding anomalies scoring, it is considered that the cluster could be considered an input, where cluster 2 represents data that it is definitively anomaly and cluster 1 represents data with a slightly lower confidence. This information was included in the files *A1_test_KMean_PCA_orig_feat* and *A1_test_KMean_VT_orig_feat*.

5 Selected output

It was not possible to select a model by its similarity with the validation dataset (*Table 3*). Instead, the selection was made by rejection: k-means models are discarded because they are just predicting as anomalies traffic with a large duration; on the other hand, iForest models based in PCA only are predicting traffic through ICMP protocol. In the remainder options, iForest with a contamination parameter set in 0.015 offers a slightly best balance between the protocols of the anomalies detected. In the *Table 4* is presented the top senders and receivers for the anomalies detected by this model.

Top senders		Top receivers	
Src IP Address	**TotalBytes (GB)**	**Src IP Address**	**TotalBytes (GB)**
224.134.91.164	11.09	160.108.108.57	1.72
146.113.93.10	1.20	202.18.114.51	1.43
121.99.92.102	1.05	205.224.112.53	1.21
192.31.93.182	0.31	169.17.118.47	1.14
150.83.100.189	0.30	175.225.124.41	1.13
145.178.112.177	0.29	154.195.93.62	1.07
144.181.96.193	0.29	228.90.27.6	1.05
202.194.98.193	0.29	176.221.130.153	0.95
180.102.106.169	0.29	141.111.103.62	0.85
157.193.102.186	0.29	195.231.110.55	0.84

Table 4 Top ten senders and receivers by TotalBytes

It is not straightforward to create a narrative for the attacks given the uncertainty in the anomalies detected. From the context of the problem it is known that the malicious traffic corresponds to Botnet operations. Even though, it is required additional information such as the DNS IP Addresses to reconstruct a proper attack narrative. Even so, *Table 5* presents a sample of a consolidation of the NetFlow traffic classified as anomalies.

Src IP Address | Protocol	**Start**	**End**	**# DestIP Address**
224.134.91.164	**01-12 15:56:55**	**08-12 10:09:00**	**595**
icmp	01-12 15:57:08	08-12 10:09:00	79
tcp	01-12 15:56:55	08-12 02:45:18	418
udp	01-12 16:21:00	08-12 08:35:37	98
146.113.93.10	**07-12 17:44:17**	**10-12 15:05:55**	**430**
tcp	07-12 17:44:17	10-12 15:05:55	430
121.99.92.102	**03-12 21:32:38**	**08-12 10:08:58**	**31**
tcp	03-12 21:32:38	08-12 10:08:58	29
udp	06-12 05:41:37	07-12 17:18:10	2
192.31.93.182	**09-12 15:17:45**	**10-12 14:13:40**	**752**
icmp	09-12 15:43:04	09-12 19:18:08	734
tcp	10-12 14:11:49	10-12 14:12:55	7
udp	09-12 15:17:45	10-12 14:13:40	11
150.83.100.189	**04-12 14:17:45**	**04-12 17:11:39**	**484**
icmp	04-12 14:43:04	04-12 17:11:39	474
udp	04-12 14:17:45	04-12 17:06:17	10

Table 5 Attack narrative reconstruction

6 Conclusion and discussion

The method worked best was iForest under Variance Threshold. It is not clear why Principal Component Analysis, even with its high dimension reduction, is generating less confident results. A potential explanation for that issue is that the reduction operation it is not suitable for an anomaly detection task because they tend to absorb the specific anomaly variability. Regarding k-means, they performance tends to be very rigid (same results under PCA and VA). This lack of flexibility difficult to tune the model to address the task.

Respect to iForest model, this method worked better in this context because allow some grade of flexibility to search the anomalies in the data. Even though, the results seem not to be perfectly accurate, as the set of anomalies detected has different metrics compared to validation set. It is considered that this issue is more related with the feature generation task rather than the model performance itself.

Finally, it is worth to mention that anomalies detection under an unsupervised machine learning scenario results in a challenging task. On the one hand, there is not available all the methods and techniques that exists in the supervised approach. On the other hand, it demands a better knowledge of the problem context to understand which the output should be.

Task 2

7 Feature selection and model

7.1 Feature selection

For this task it was chosen a different mode to process the traffic data. The NetFlow registers were grouped by the field SrcIPAddress, resulting in a more compact file. Most importantly, this election was very convenient for the purpose of the task, which is trying to predict if an IP address (or its NetFlow traffic) is associated with Botnets. As it will reviewed in section 7.2, it was obtained a high accuracy with the help of this operation.

In summary, the pre-processing and feature selection subtasks were:

- Similar to Task 1, they were added the following features: Packets by Second, Total Bytes by Second and Source Bytes by Second.

- The multi-value label of the dataset was simplified into a binary label named "Botnet". Basically, it was filtered the original values which contained the word "Botnet".
- It was grouped the dataset by the field SrcIPAddress under an average operation. They only remain the numerical features.

All these subtasks are contained in the file *A2_preprocessing*. The *Figure 5* shows an overview of the train dataset after to apply these subtasks.

SrcIPAddress	Duration	TotalPackets	TotalBytes	SourceBytes	Botnet	PacketsBySec	SourceBytesBySec	TotalBytesBySec
124.232.153.174	0.104518	2.286232	149.264493	78.702899	0.0	13882.919749	431491.252983	833081.520768
147.102.96.196	35.829232	13.564209	9173.177942	1398.448550	1.0	1662.120009	48544.183819	237982.640615
147.115.98.191	15.056139	16.217323	11788.204134	1437.707480	1.0	1773.958664	52080.610326	257499.951242
147.119.98.190	36.468520	7.010240	2880.259186	917.122696	1.0	1635.948241	47796.257082	237380.888041
147.124.93.156	14.920804	5.567289	2039.830527	593.554574	1.0	1751.074834	50859.497350	255220.597559

Figure 5 Train dataset for Task 2

7.2 Model

Due to its simplicity, it was chosen the model Logistic Regression. Under this model was obtained an accuracy of 0.91. It was utilised the default parameters from the Skit Learn method. The *Figure 6* shows a sample of the results. This model is available in the file *A2_model_Logistic_Regression_Adversarial_Sample* as it contains both the model and the generation of one adversarial sample for this model (section 8).

	SrcIPAddress	Botnet	Predicted
0	122.226.12.150	0.0	1.0
1	147.101.94.182	1.0	1.0
2	147.113.94.198	1.0	1.0
3	147.114.98.192	1.0	1.0
4	147.122.97.192	1.0	1.0

Figure 6 Task 2 prediction sample

8 Adversarial sample

8.1 Generation

Diverse techniques exist for adversarial machine learning [4]. In this project, it was requested to utilise an evasion attack, which it assumes the possibility of modifying the data rather than the model. One the simplest solutions is based in gradient descent and it is named Fast Gradient Sign Method (FGSM). In this method, a new adversarial sample is generated as follow:

$$x' \leftarrow x + \varepsilon \cdot sign\left(\frac{\partial loss_{true}}{\partial x}\right)$$

The implementation of this method is relatively straightforward [5]. For these tasks it was generated an adversarial sample for just one Source IP. From *Figure 6* it will be chosen the row 1, where the Source IP Address **147.101.94.182** is correctly predicted as Botnet. The *Figure 7* shows the code which generates an adversarial sample for this row. A couple of notes regarding this implementation:

- epsilon was defined with a large value because the data is not normalized: it is considered unnormalized is more practical for "backpropagate" the epsilon to the NetFlow traffic.
- The sign of direction was directly set as 1: It was not possible to determine the gradient function given a Skit Learn model (there is not direct access to the function weights).

```
IP = X_test.iloc[[1]]
Label = Y_test.iloc[[1]]
epsilon = 50
direction = 1
adversarial_sample = IP.values + epsilon * direction
```

Figure 7 Adversarial sample

Finally, in *Figure 8* it is presented the prediction of the model for this new adversarial sample. As we can verify, now the model does not predict the Source IP as Botnet. The accuracy of the model decreases to 0.89.

	SrcIPAddress	Botnet	Predicted
0	122.226.12.150	0.0	0.0
1	147.101.94.182	1.0	0.0
2	147.113.94.198	1.0	1.0
3	147.114.98.192	1.0	1.0
4	147.122.97.192	1.0	1.0

Figure 8 Model prediction for adversarial sample

8.2 Traffic perturbation

Unfortunately, this task was unsuccessful. The main challenge was to introduce the epsilon generated in section 8.1 (epsilon = 50) in the NetFlow traffic given under a group operation. The first attempt consisted in just use this epsilon in all the traffic related to the target IP Source. This idea did not work because the model has generated features as TotalBytes by seconds that are created from the division of another two features. The introduction of the epsilon in all the traffic distorted these types of metrics. The second attempt considered introducing the effect of this epsilon accumulated in just one traffic row. This attempt is registered in the file *A2_preprocessing-perturb-traffic*. Even when it was generated a perturbation very similar to the target, again, the generated features are "defending" the model against this attack. The *Figure 9* and *Figure 10* show how similar is the distortion generated.

	Duration	TotalPackets	TotalBytes	SourceBytes	PacketsBySec	SourceBytesBySec	TotalBytesBySec
0	0.068330	2.099315	125.989726	68.424658	10164.267421	313196.425928	609901.995847
1	87.038085	56.736226	2669.486505	1010.705275	1546.108699	43839.455999	211946.322906
2	35.870162	13.585749	9117.484682	1401.067273	1661.905285	48534.566173	237852.833934

Figure 9 Distortion generated directly in the groped data

	SrcIPAddress	Duration	TotalPackets	TotalBytes	SourceBytes	Botnet	PacketsBySec	SourceBytesBySec	TotalBytesBySec
0	122.226.12.150	0.068330	2.099315	125.989726	68.424658	0.0	10164.267421	313196.425928	609901.995847
1	147.101.94.182	87.038085	56.736226	2669.486505	1010.705275	1.0	1495.671372	43775.022442	211851.928670
2	147.113.94.198	35.870162	13.585749	9117.484682	1401.067273	1.0	1661.905285	48534.566173	237852.833934

Figure 10 Distortion generated in the NetFlow traffic

In this case it is assumed that a potential attacker does not have access to the grouping operation. If not, the attack could be carried out easily.

9 Conclusion and discussion

Regarding the adversarial technique utilised, Fast Gradient Sign Method it does not produce the minimal adversarial perturbations. This was clear in this task, where it was necessary to introduce a huge epsilon to the data. For some features, as Duration, this value represented a complete change in the magnitude of the value. On the other hand, as the dataset utilised in the machine learning model was generated grouping the NetFlow traffic, it turned very difficult to introduce this modification in the original data.

Comparing this task with a computer vision project, one important difference is that in that field there are more examples and applications. It is a more explored field compared with network traffic. On the other hand, to introduce adversarial samples in traffic data should be more difficult in practice than in computer vision. For example, should not be easy to fool a firewall with traffic that have a huge perturbation. In computer vision, in general there is not that kind of filters.

10 Bibliography

[1] M. Brems, "A One-Stop Shop for Principal Component Analysis," Towards Data Science, 18 April 2018. [Online]. Available: https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c. [Accessed 18 October 2020].

[2] S. Erfani, "An Introduction to Anomaly Detection," University of Melbourne - School of Computing and Information Systems, 2020.

[3] S. Erfani, "Clustering and Density-based Anomaly Detection," University of Melbourne - School of Computing and Information Systems.

[4] Y. Han, "Adversarial Machine Learning – Vulnerabilities," University of Melbourne - School of Computing and Information Systems, 2020.

[5] K. Tsui, "Perhaps the Simplest Introduction of Adversarial Examples Ever," Towards Data Science, 21 August 2018. [Online]. Available: https://towardsdatascience.com/perhaps-the-simplest-introduction-of-adversarial-examples-ever-c0839a759b8d. [Accessed 15 October 2020].