COMP90073 Security Analytics, Semester 2 2020

Project 2: Machine Learning based cyberattack detection

Name: Diego Aranda Villarreal
Student ID: 992038

October 20th, 2020

Tabla de contenido

1 Introduction *3*

2 Task 1 – Anomaly detection using Machine Learning algorithms *3*

2.1 Overview of the dataset **3**

2.1.1 Loading the dataset 3

2.1.2 Dataset 3

2.1.3 High level view of the dataset 3

2.2 Feature generation and selection **8**

2.2.1 Feature generation 9

2.3 Anomaly Detection **10**

2.3.1 Data pre-processing 10

2.3.2 Experiment design 10

2.3.3 Anomaly Detection Models 11

2.4 Attack description and discussion **17**

3 Task 2 – Generating Adversarial Samples *18*

3.1 Model and feature selection **18**

3.2 Fast Gradient Descent Method **19**

3.3 Attacking the LR model **19**

3.4 Conclusion and discussion **20**

References *22*

Appendix 1 – Attackers and Victims for Task 1 .... *23*

Appendix 2 – ".csv" and ".ipynb" files details *25*

Appendix 3 – Predicted botnet IPs for Task 2 .... *26*

1 Introduction

The following report analyzes the behavior of a net flow archive by the means of unsupervised learning methods, in order to find potential cyberattacks in the form of a botnet architecture. Also, a supervised learning model is trained and adversarial samples are generated to check whether the model is susceptible to evasion attack.

2 Task 1 – Anomaly detection using Machine Learning algorithms

2.1 Overview of the dataset

2.1.1 Loading the dataset

The dataset is formed by three parts: training set, test set and validation set. We introduced the datasets into Splunk in order to have an overview of the different components of the data. The file was ingested by putting the file into the PCAP folder of Splunk that contains ".csv" transformed files. For loading the dataset into Jupyter Notebook, we used the reading function of the pandas framework provided by the corresponding library.

2.1.2 Dataset

Given that the dataset didn't contain headers for each column of data, we used the Extract Fields functionality provided by Splunk in order to put the names of the 14 fields and for our queries to be easier to construct. The training set contained 13,882,035 transactions and test set contained 1,053,845 transactions.

2.1.3 High level view of the dataset

We performed an exploratory analysis over the training and test datasets by using Splunk queries. The aim was to find patterns related to conversations between IP addresses, the use of protocols and peaks of bytes and packet transfers.

Figure 1 shows that most used protocol in the training set network traffic corresponded to UDP, TCP and ICMP respectively. In turn, test set shows that activity increased for TCP and ICMP, leaving TCP as the third most used protocol.

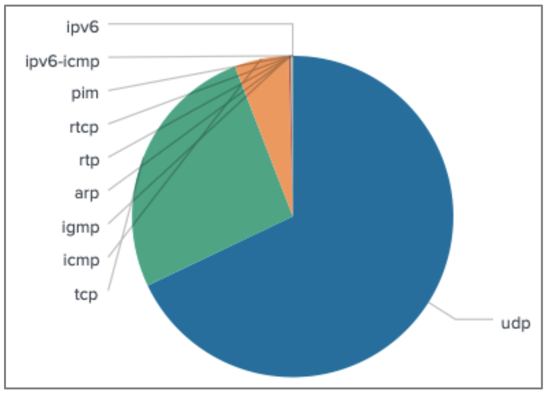


Figure 1 – Top protocols training data

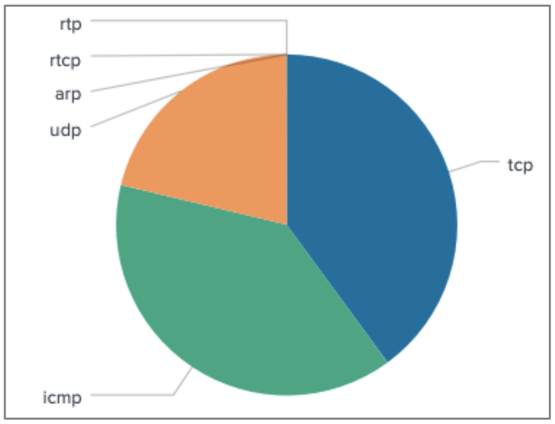


Figure 2 – Top protocols test data

By following the same patterns as before, Figure 3 shows that most used destination ports corresponds to 53, 80 and 13363, which are commonly used for the top protocols. Similarly, Figure 4 shows ports 53, 25 and 22.

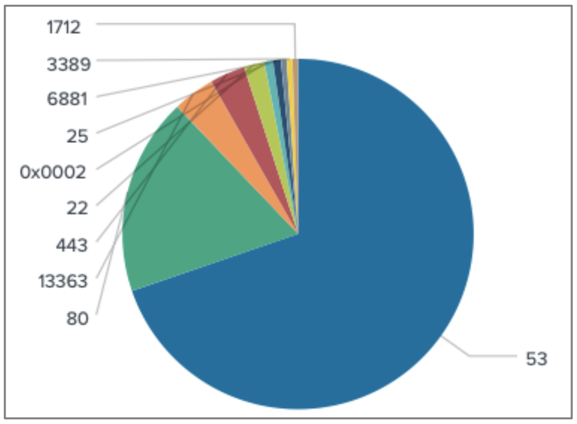


Figure 3 – Top destination ports training data

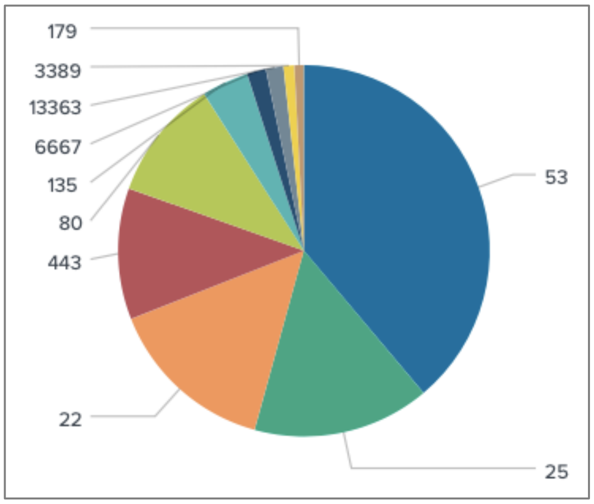


Figure 4 – Top destination ports test data

Regarding the conversation between hosts, Figure 5 shows that several IPs attempted to establish connection with address 125.189.87.2. In Figure 6, it can be seen that IP 224.134.91.164 attempted to connect with several destination IPs. Both of these behaviors are to be considered suspicious for our former analysis.

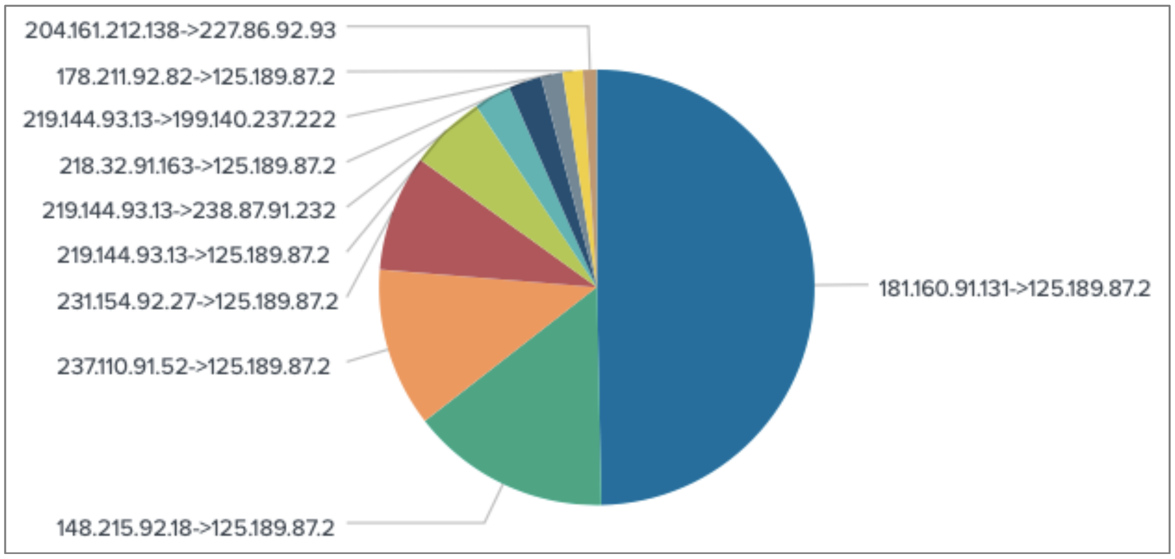


Figure 5 – Top conversations training data

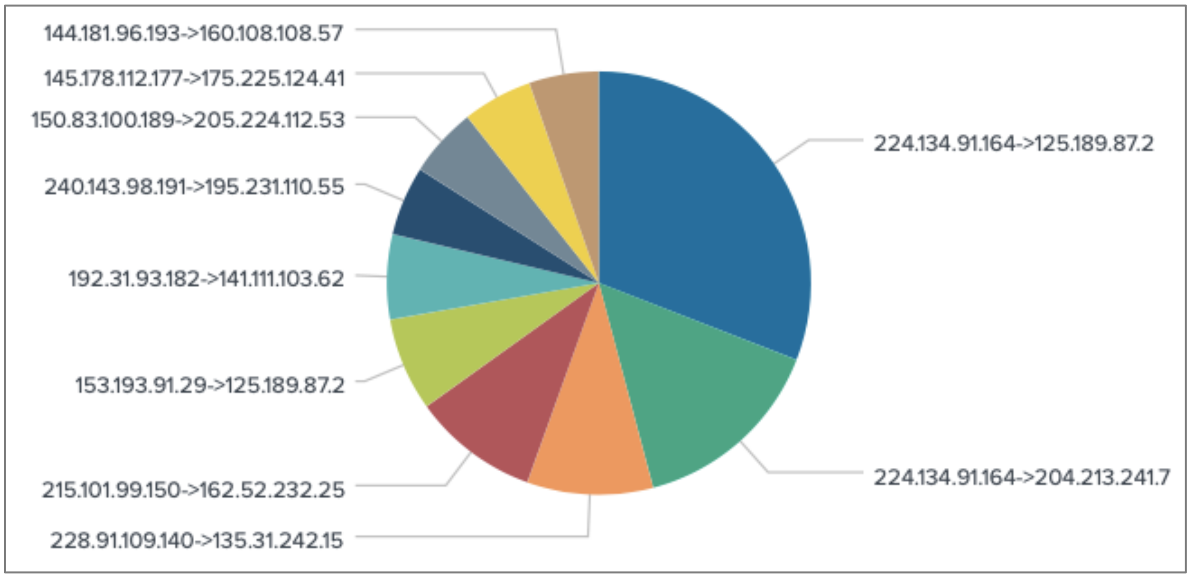


Figure 6 – Top conversations test data

Top source and destination IPs are shown in Figure 7 and Figure 8. Moreover, it can be seen that 224.134.91.164 appears again as top source IP, and 125.189.87.2 appears to be the top sender, which could be related to suspicious behavior in the network.

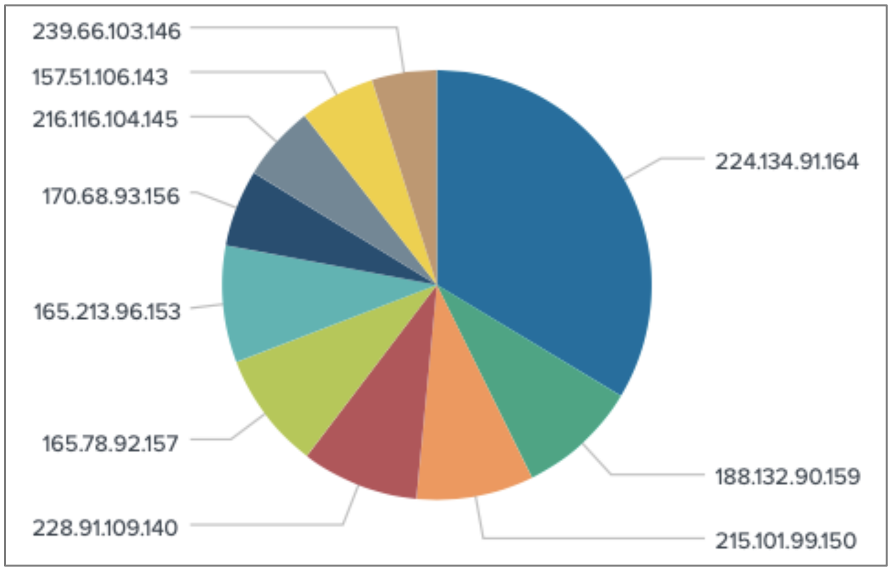


Figure 7 – Top source IPs for test data

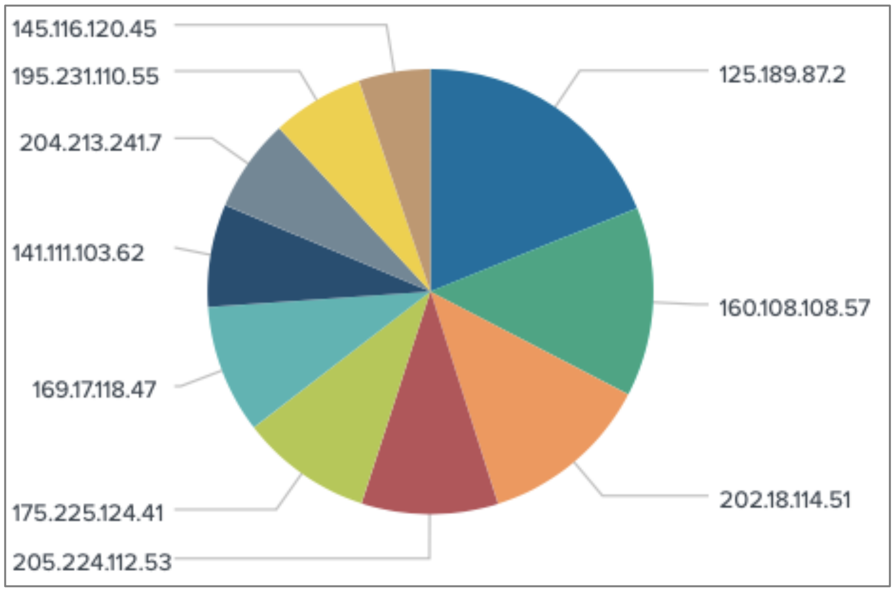


Figure 8 – Top destination IPs for test data

By taking into account the transfer of bytes, we can see in Figure 9 and Figure 11 that 224.134.91.164 was again involved in suspicious behavior at several point of the time space. In Figure 10, we can observe that there might be suspicious transfers of data that might be aligned with bytes peaks from Figure 11.

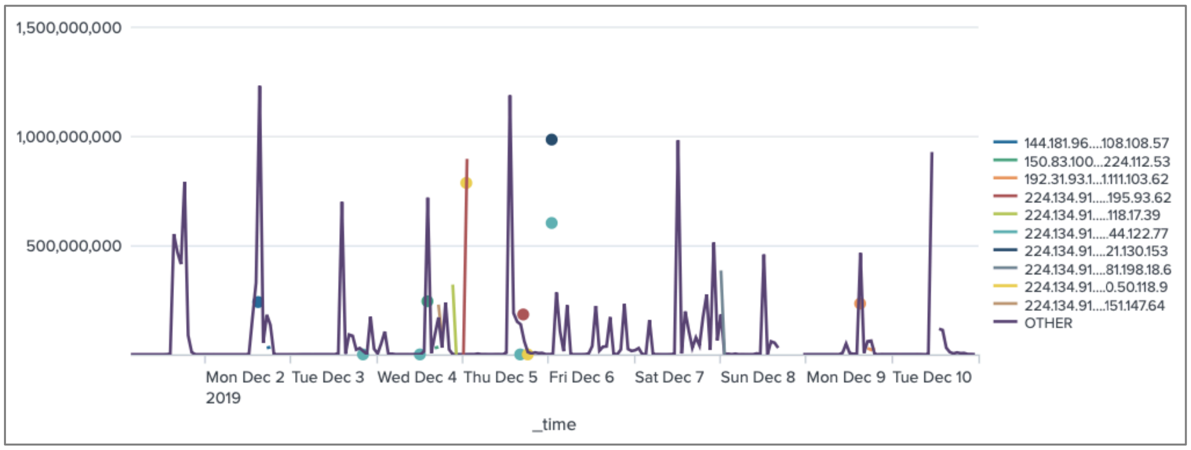


Figure 9 – Bytes transferred by conversation for test data

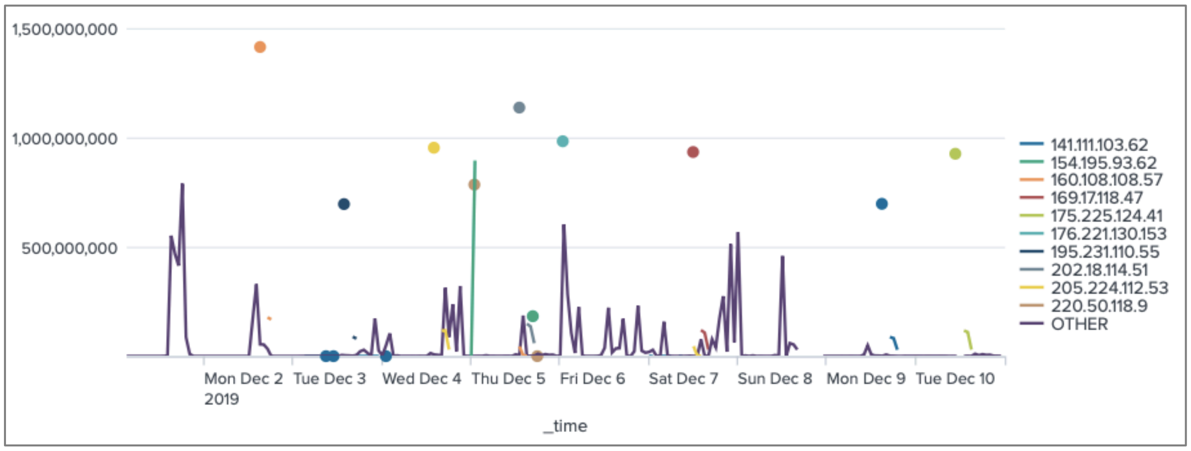


Figure 10 – Bytes transferred to destination IP for test data

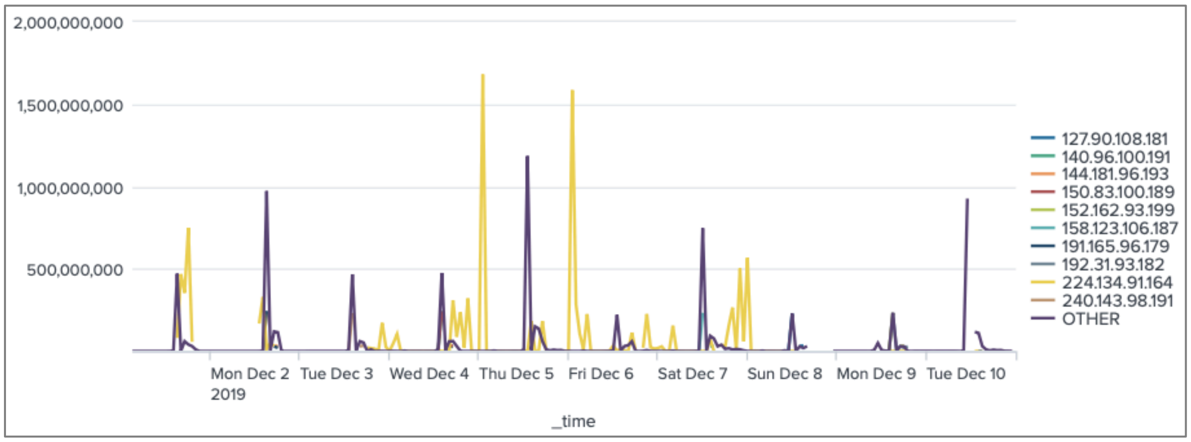


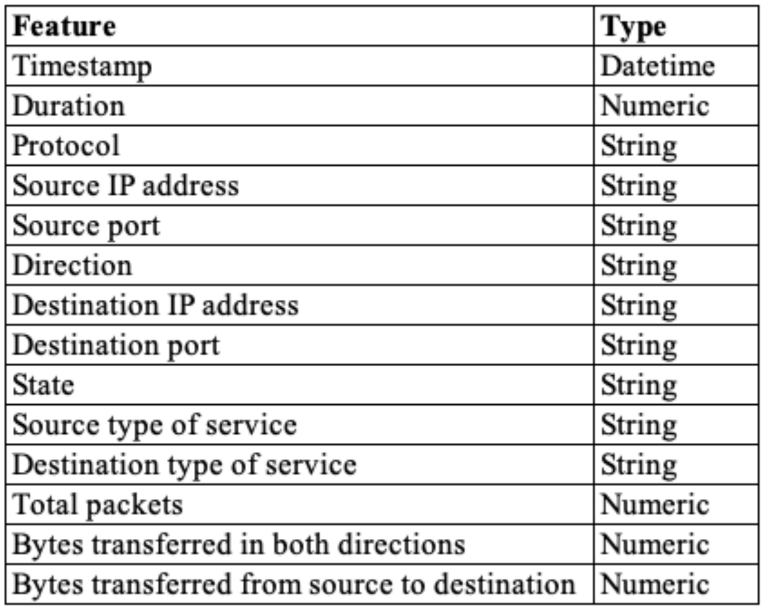
Figure 11 – Bytes transferred by source IP for test data

2.2 Feature generation and selection

According to our overview, we consider that the test dataset might present suspicious behavior in terms of traffic between source and destination IPs, the amounts of bytes and packets transferred, the source and destination ports, the protocols of communication and the amount of conversations.

The analysis of this report will be focused on proposing a set of features to be extracted from the datasets in order to detect anomalies in network traffic, by the means of unsupervised machine learning models. We used the Pandas library from python in order to generate the features to be used for our analysis. The list of basic extracted features was can be seen in Table 1.

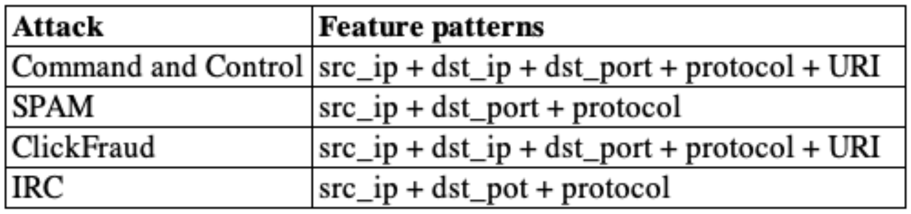
Table 1 – Basic features for Assignment 2



Feature	Type
Timestamp	Datetime
Duration	Numeric
Protocol	String
Source IP address	String
Source port	String
Direction	String
Destination IP address	String
Destination port	String
State	String
Source type of service	String
Destination type of service	String
Total packets	Numeric
Bytes transferred in both directions	Numeric
Bytes transferred from source to destination	Numeric

Regarding Assignment 1, the basic features were given directly by ingesting the PCAP file into the required system location, and also the feature URI was extracted by using the extract fields functionality from Splunk. The discovery of four patterns of attack was attained by exploring the data by the guidance of the requirements of the report.

Table 2 – Feature patterns of attack for Assignment 1



Attack	Feature patterns
Command and Control	src_ip + dst_ip + dst_port + protocol + URI
SPAM	src_ip + dst_port + protocol
ClickFraud	src_ip + dst_ip + dst_port + protocol + URI
IRC	src_ip + dst_pot + protocol

2.2.1 Feature generation

As defined by Sperotto et al. (2010), a flow can be defined by the combination of IP source, IP destination, port source, port destination and protocol of communication. Our analysis proposes to use a modified version of the flow ID as a combination of the IP source, the port source, the direction, the IP destination and the port destination. In addition, we consider to add numeric features based on the basic ones delivered by the dataset, and maintain a portion of the categorical features. The following Table 3 presents our 10 selected features for analysis.

Table 3 – Features generated for our analysis

Feature	Details
flow ID	Src_ip + src_port + direction + dst_ip + dst_port
protocol	Communication protocol
state	Categories for state of the network flow
pps	Packets per second in one direction
bps_oneway	Bytes per packet in one direction
bpp_oneway	Packets per second in one direction
bps_twoway	Bytes per packet in two directions
bpp_twoway	Packets per second in two directions
src_type_service	Source type of service
dst_type_service	Destination type of service

2.3 Anomaly Detection

Anomaly detection aims to find data patterns that deviate from normal behavior. This unsupervised method assumes for the data to be unlabeled, thus normal class transaction should be much greater than anomalous data (Chandola et al., 2009). We will be analyzing test data by checking the potential outliers which could represent anomalous behavior in the network.

2.3.1 Data pre-processing

Data was firstly explored in Splunk, for which basic 14 features were extracted. Then, data was loaded into Jupyter Notebook to be preprocessed. Then, features were generated by using Pandas library common functions for each dataset.

Given our IT resources constraints, we considered to use Label Encoder class from Sklearn Proprocessing library to encode categorical features, as when we tried to use OneHotEncoder class we got a memory error.

2.3.2 Experiment design

We used two anomaly detection methods: iForest and Local Outlier Factor (LOF). Each model was run with three different feature sets, firstly with our 10 generated features, and then using two feature selection methods, namely Incremental Principal Components Analysis using 5 components, a PCA version for large datasets which adds more efficiency for constrained machines, and Variance Threshold method, which aims to remove low variance features. Both feature selection methods are of unsupervised nature.

Table 4 – Models' performances

	iForest				LOF		
	Contamination	**Training time**	**Test time**	**Anomalies**	**Contamination**	**Test time**	**Anomalies**
Generated Features	0.01	9min 6s	28.5s	4,763	0.001	1min 5s	1,054
	0.02	9min 3s	28.1s	187,175	0.01	1min 6s	10,539
	0.1	9min 7s	29.2s	255,826	0.1	1min 4s	105,384
	0.25	9min 11s	28.2s	948,599	0.2	1min 7s	210,769
IPCA	0.01	8min 14s	27.5s	27,174	0.001	31.1s	1,054
	0.02	8min 9s	24.9s	36,106	0.01	31.1s	10,539
	0.1	8min 9s	25.1s	497,948	0.1	31.7s	105,385
	0.25	8min 10s	25s	870,699	0.2	31.2s	210,769
VT	0.01	8min 51s	27.4s	11,047	0.001	59.6s	1,054
	0.02	8min 38s	27.8s	154,436	0.01	58.9s	10,539
	0.1	8min 44s	27.9s	368,292	0.1	1min 3s	105,385
	0.25	8min 40s	27.3s	1,002,942	0.2	58.2s	210,769

2.3.3 Anomaly Detection Models

For this section we present 1 result for each anomaly detection model taking the best run from our feature selection techniques. Our choice is based on inspection of the two generated ".csv" files by comparing them against our overview results from Section 2.1.3.

2.3.3.1 iForest

The aim of the iForest model is to isolate records in contrast to profiling normal observations. The algorithm split the records by choosing a feature randomly and then splits again by randomly picking a value between the maximum and the minimum values with respect to the chosen feature (Liu et. al. 2008). It delivers results though an ensemble of trees, which makes it highly scalable, fast and interpretable (Erfani 2020).

2.3.3.1.1 Experimental design

We considered to tune only one parameter of the Sklearn iForest model, which is contamination, and the selected values can be seen at Table 4. Contamination parameter shows the proportion of observations to be considered as outliers.

2.3.3.1.2 Scoring and threshold

The contamination parameter is to determine the threshold with which iForest decides whether an observation is an anomaly or not. The Sklearn method returns anomalies as -1 and normal observations as 1 for scoring based on the decision function defined by Liu et. al. (2008). We mapped those values as 1 for anomalies and 0 for normal observations.

2.3.3.1.3 iForest results

We choose to show results for the IPCA model with 1% contamination, as most of the src_ip and dst_ip retrieved were present in our top metrics from the Splunk overview as seen in Figure 12, 13 and 14. Moreover, top conversations were also overlapped. The main protocol used was ICMP (Figure 15). Furthermore, one can see that from Figure 16 that our model was able to separate anomalies from normal points by using IPCA method.

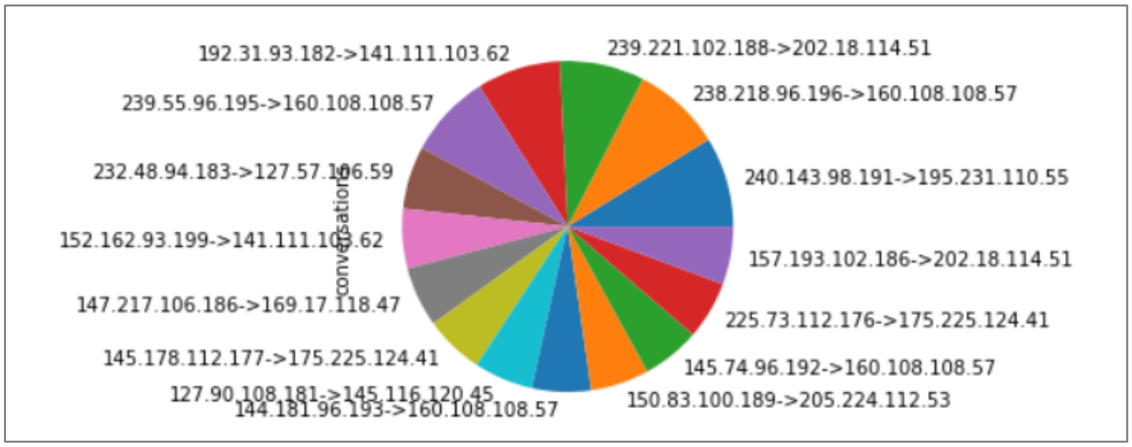


Figure 12 – Top conversations iForest

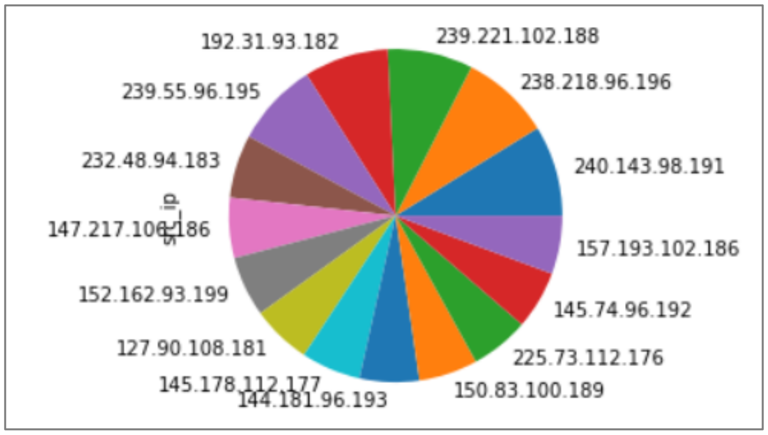


Figure 13 – Top src_ip iForest

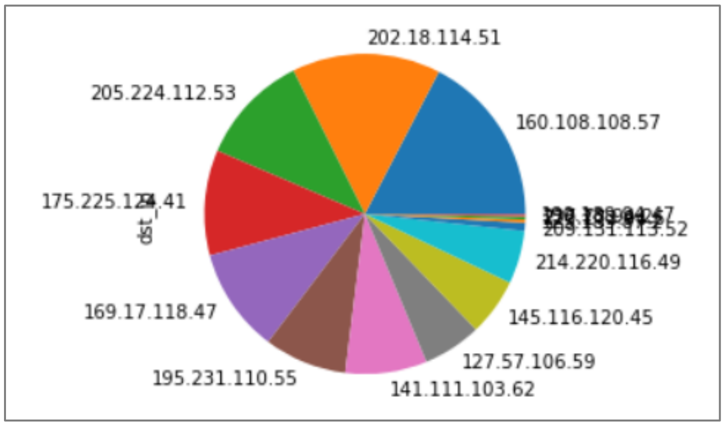


Figure 14 – Top dst_ip iForest

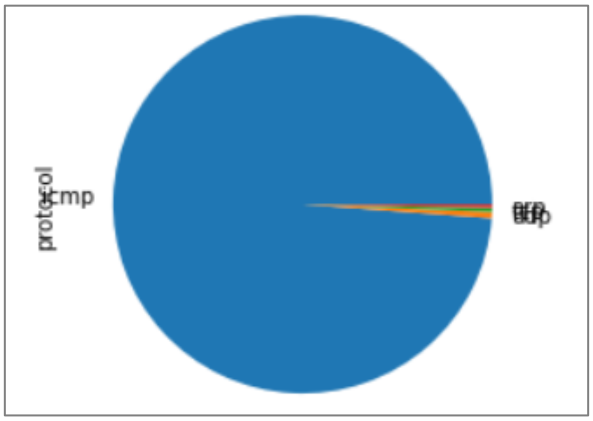


Figure 15 – Top protocol iForest

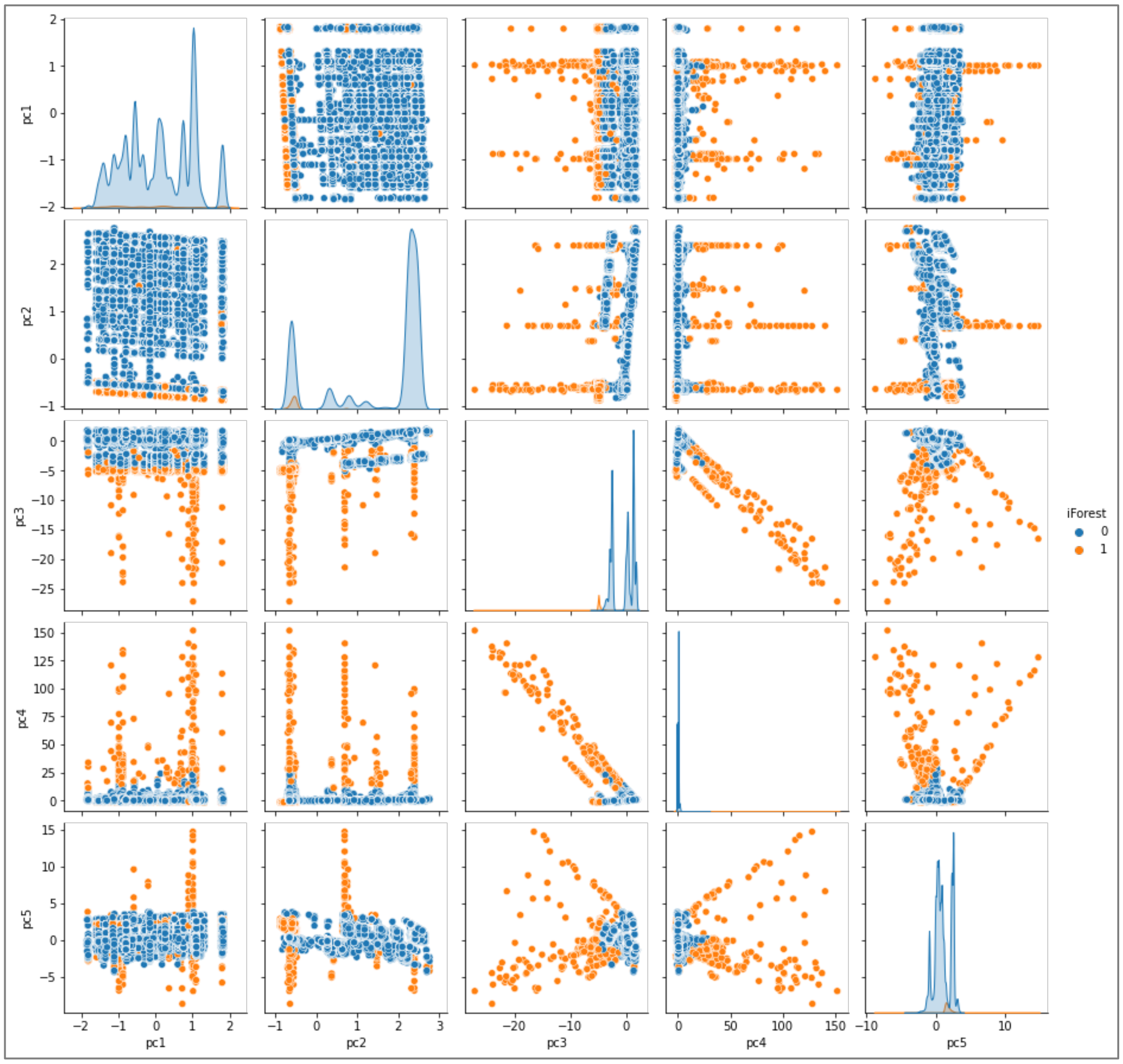


Figure 16 – iForest IPCA correlations for 5 components

2.3.3.2 LOF

LOF is a local proximity based algorithm for calculating anomalies within a dataset, which considers the local relative density of observations with respect to its neighbors to determine the degree of anomalous behavior. Its local components is related to the degree to which an observation is isolated with respect to its nearest neighbors (Breunig et al. 2000).

2.3.3.2.1 Experimental design

We considered to use two parameters for LOF. We set the contamination levels as shown in Table 4 and the k-neighbors as 35 arbitrarily. Contamination parameter shows the proportion of observations to be considered as outliers. In order to determine the amount of k-neighbors, we considered a value greater to the number of features of our dataset and assessed the differences in the reported IPs to select one result (Breunig et al. 2000).

2.3.3.2.2 Scoring and threshold

LOF from Sklearn maps anomalies as -1 and normal observations as 1. We mapped those values as 1 for anomalies and 0 for normal observations. Moreover, the identification of anomalies is influenced by the value of k-neighbors, which in this case was selected as a values greater than the amount of features as part of the threshold.

2.3.3.2.3 LOF results

We choose to show results for the IPCA model with 1% contamination, as most of the src_ip and dst_ip retrieved were present in our top metrics from the Splunk overview as seen in Figure 17, 18 and 19. The main protocol used was ICMP (Figure 20). Moreover, top conversations were also overlapped. Furthermore, one can see that from Figure 21 that our model was able to separate anomalies from normal points by using IPCA method.

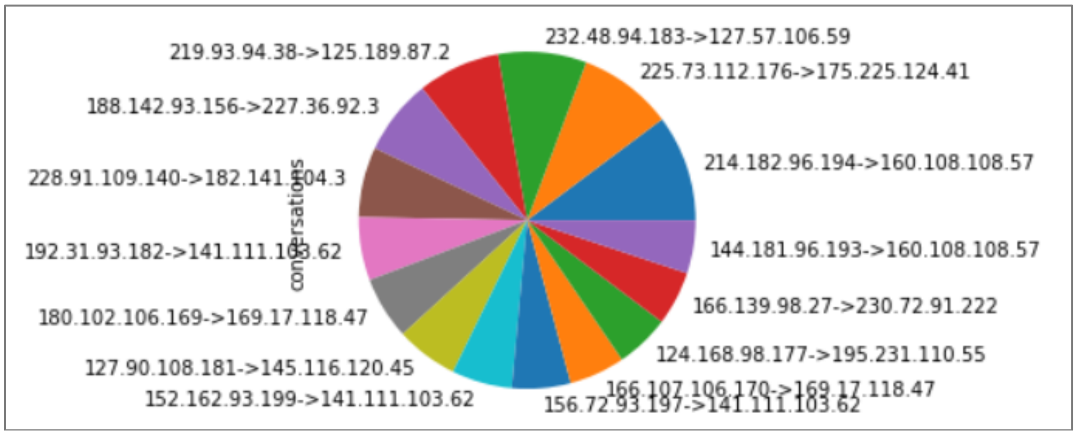


Figure 17 – Top conversations LOF

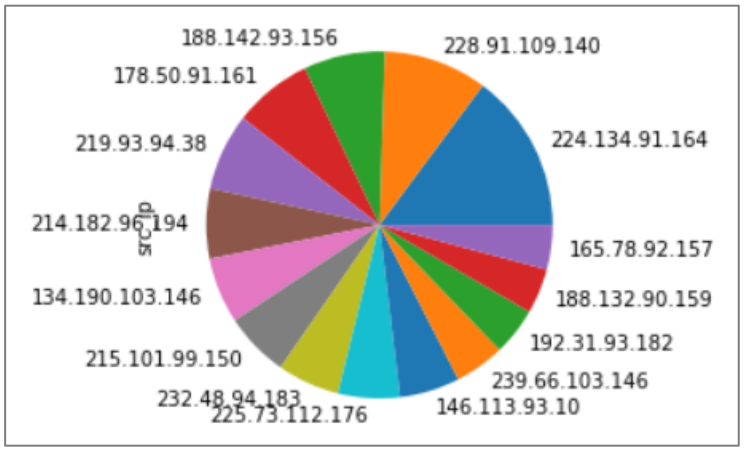


Figure 18 - Top src_ip LOF

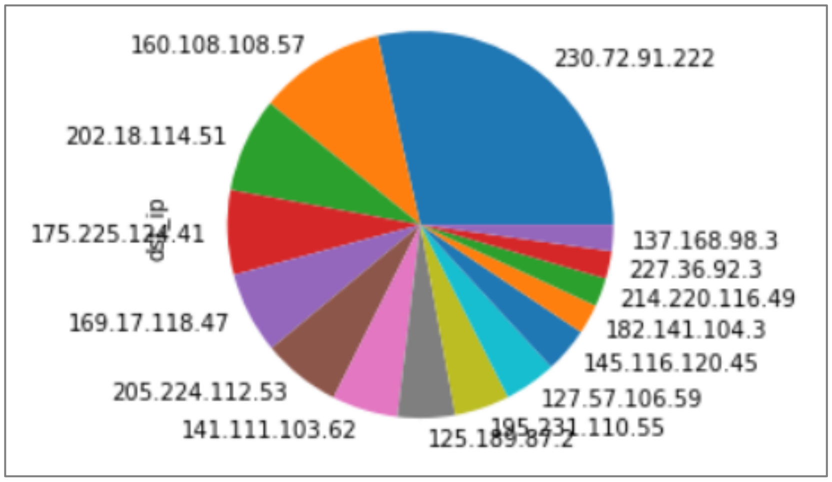


Figure 19 - Top dst_ip LOF

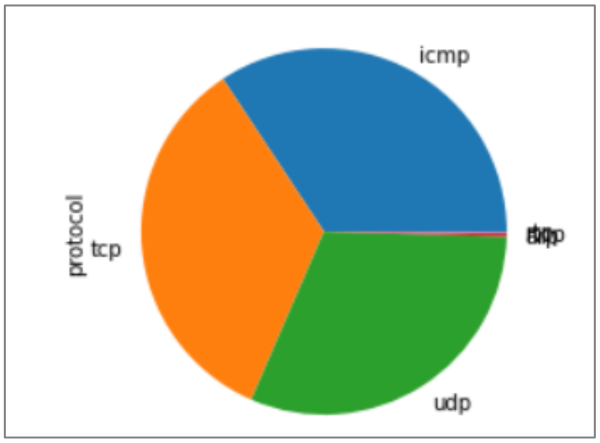


Figure 20 – Top protocol LOF

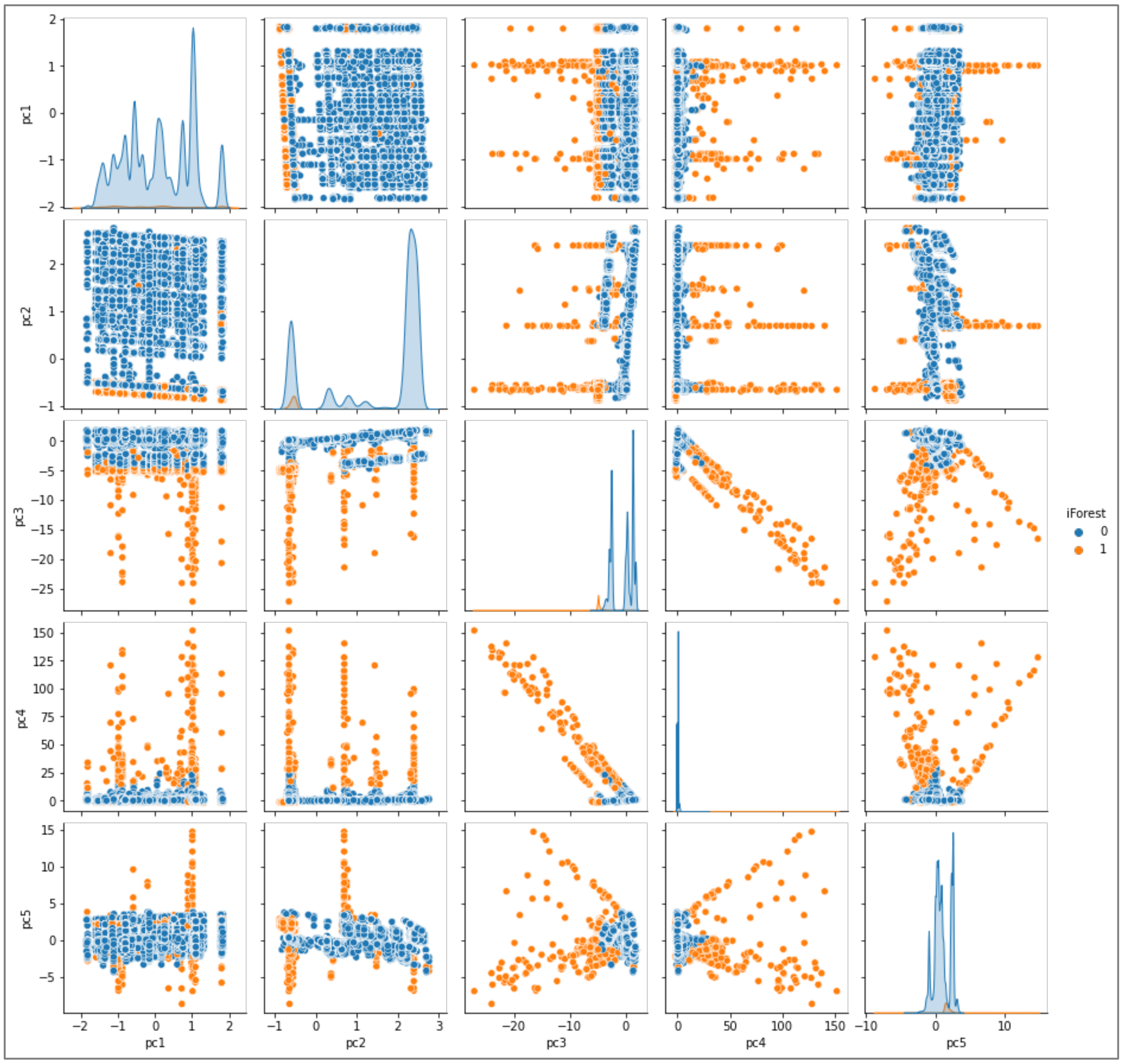


Figure 21 – iForest IPCA correlations for 5 components

2.4 Attack description and discussion

We consider for the iForest model to be the more accurate as it considers a more reduced amount of src_ip and dst_ip, and contains all the top conversations, src_ip and dst_ip from our overview analysis. We consider a list of 79 attackers and 101 victims which can be found in Appendix 1. We considered for 27,174 records to be anomalous, and for the majority of the attacks to be a ICMP based DDoS attack, which started at 2:48:22 PM and ended by 6:51:44 PM on 10/12/2012. Our final "test_result_iforest_001_ipca.csv" file contains the detail of all the records involved and the stream ID of each transaction (stream ID starts at 1). We also include 5 more ".csv" files as proof of our work, in addition to the required Jupyter Notebook files (all explained in Appendix 2).

We considered that iForest worked better than LOF tended to show a much greater amount of unique src_ip and dst_ip values, and although much of the top conversations from LOF overlapped with our overview analysis, LOF tended to capture much noise in comparison with iForest. One of the key issues with LOF is that our IT constraints prevented us from using the training set to be combined with the test set to achieve more accurate results for detecting anomalies. Computer memory should be considered for machine learning models to be scalable and for real-time detection tasks.

3 Task 2 – Generating Adversarial Samples

Adversarial Machine Learning is a novel method by which cyber-attackers may leverage machine learning to modify the inputs to trick machine classification tasks (Han 2020).

3.1 Model and feature selection

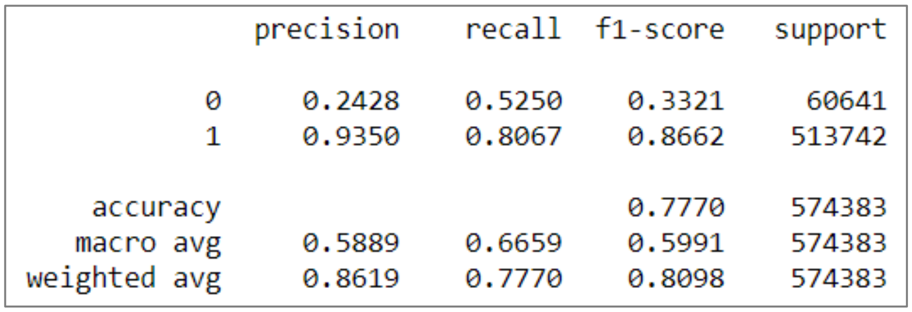
We selected Logistic Regression (LR) model in order to generate the adversarial samples. LR is a binary classification model that attempts to discriminate between classes by calculating the probability of each class given the inputs (Frermann 2020). Given this scenario, LR model uses gradient descent method to arrive to the following loss function:

$$[\sigma(\theta^T x) - y] \times x_j \qquad (1)$$

In this case, the sigmoid function represents the probability estimate by which the LR model will decide the class of the input according to the decision boundary, y represents the true value of the label and x_j represents the input.

We selected a subset of 6 numerical features of our original 10, which is comprised of pps, bps_oneway, bpp_oneway, bps_twoway and bpp_twoway. This first selection was made for avoiding using categorical features and to increase the accuracy of the LR model when using the test set. The results of this are shown in Table 5.

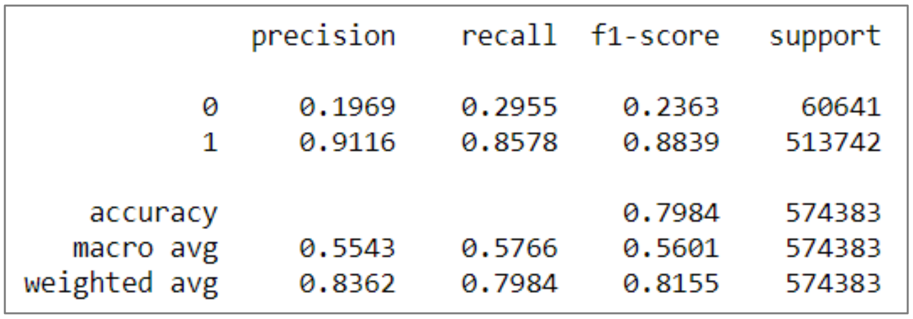
Table 5 – Accuracy LR 6 features



```
              precision    recall  f1-score   support

           0     0.2428    0.5250    0.3321     60641
           1     0.9350    0.8067    0.8662    513742

    accuracy                         0.7770    574383
   macro avg     0.5889    0.6659    0.5991    574383
weighted avg     0.8619    0.7770    0.8098    574383
```

Then, we applied Mutual Information (MI) in order to select the 3 best features out of our subset. MI aims to measure the reduction of uncertainty for one input given a known value of the output. The selected features corresponded to bps_twoway, pps and bps_oneway. The results after using MI can be seen in Table 6.

Table 6 – Accuracy LR 3 features



```
              precision    recall  f1-score   support

           0     0.1969    0.2955    0.2363     60641
           1     0.9116    0.8578    0.8839    513742

    accuracy                         0.7984    574383
   macro avg     0.5543    0.5766    0.5601    574383
weighted avg     0.8362    0.7984    0.8155    574383
```

3.2 Fast Gradient Descent Method

Our selected technique for generating the adversarial samples corresponds to Fast Gradient Descent Method (FGSM), by which we can perturb the inputs to induce our LR model to make mistakes. The following formula explains how to perturb the features of the input:

$$x' \leftarrow x + \varepsilon \times sign\left(\frac{\partial loss}{\partial x}\right) \qquad (2)$$

In our case x' corresponds to the adversarial samples, x is the input of test set, ε is a small number equal to 0.8 chosen arbitrarily, and the partial derivative corresponds to the gradient of cost function with respect to the input x. The calculation of the partial derivative of loss in adversarial samples goes as follows:

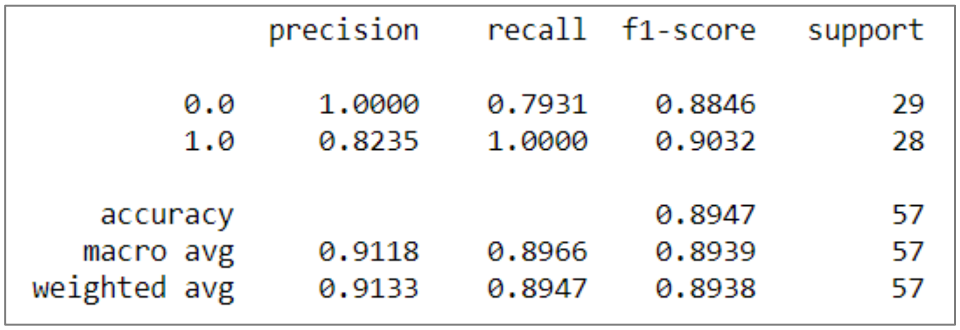
$$[\sigma(\theta^T x) - y] \times \theta \qquad (3)$$

θ represents the weights from the LR model. The detail of the functions used and the code can be seen in "advSamplesTask2.ipynb" Jupyter Notebook file.

3.3 Attacking the LR model

We chose one IP address labeled originally as botnet, for which we compressed both datasets using group by function over the src_ip feature, and then train and test the LR model over the reduced sets. The results are shown below in Table 7. The predicted botnet IPs are shown in Appendix 3.

Table 7 – Accuracy LR unique IPs



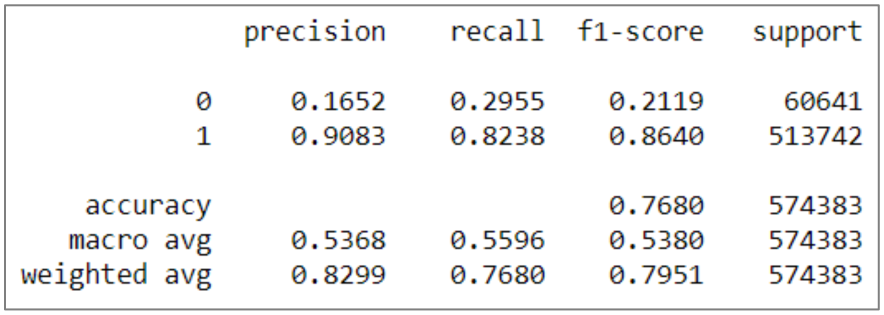
```
              precision    recall  f1-score   support

         0.0     1.0000    0.7931    0.8846        29
         1.0     0.8235    1.0000    0.9032        28

    accuracy                         0.8947        57
   macro avg     0.9118    0.8966    0.8939        57
weighted avg     0.9133    0.8947    0.8938        57
```

The chosen IP corresponds to 147.101.94.182. This IP was present in 20,035 records originally in the test set, for which 17,462 were classified as botnet, and all of its true labels corresponded to botnet class. Our steps to update the network traffic went as follows:

a. Select the 20,035 rows corresponding to 147.101.94.182 and hold the true values.
b. Calculate the partial derivative by subtracting the probability estimates from botnet class vector and the true labels, and multiply that result with the weights of the LR model.
c. Retrieve the direction of the partial derivative by using sign function.
d. Calculate adversarial samples as in (2).
e. Test the model again.

After performing the above actions, all of the predictions for 147.101.94.182 were classified as normal and the accuracy went from 79.84% to 76.80%. The final results are shown below in Table 8. Our file "task2_export.csv" holds the values that were perturbed for the 3 columns, we perturbed the scaled features.

Table 8 – Accuracy including adversarial samples



```
              precision    recall  f1-score   support

           0     0.1652    0.2955    0.2119     60641
           1     0.9083    0.8238    0.8640    513742

    accuracy                         0.7680    574383
   macro avg     0.5368    0.5596    0.5380    574383
weighted avg     0.8299    0.7680    0.7951    574383
```

3.4 Conclusion and discussion

One key difference is to be spotted between generating adversarial samples for network traffic and computer vision. Adversarial samples in network traffic may be different in terms of properties and distribution in comparison to training data, which makes the task hard for attackers as they need to understand how the training data changes over time. In this way, adversarial samples in the context of network traffic adds the element of time, because training data changes constantly and adds complexity to the attack task (Xi 2020).

In contrast, adversarial images can be created with little perturbations in the inputs when trying to break deep neural networks, which was the case of Tesla that caused a fatality because of the limitations of the Tesla Autopilot system. Thus, machine learning techniques used in computer vision, although being very complex algorithms, may still be vulnerable to attacks that need less efforts compared to generating adversarial network traffic (Xi 2020).

References

Breunig, M. M., Kriegel, H. P., Ng, R. T., & Sander, J. (2000, May). LOF: identifying density-based local outliers. In Proceedings of the 2000 ACM SIGMOD international conference on Management of data (pp. 93-104).

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. ACM computing surveys (CSUR), 41(3), 1-58.

Erfani, S. 2020, Teaching Sessions 9, COMP90073 Security Analytics, Learning materials on Canvas LMS, The University of Melbourne, viewed October 20th 2020.

Frermann, L. 2020, Teaching Sessions 7, COMP90049 Introduction to Machine Learning, Learning materials on Canvas LMS, The University of Melbourne, viewed October 20th 2020.

Han, Y. 2020, Teaching Sessions 17, COMP90073 Security Analytics, Learning materials on Canvas LMS, The University of Melbourne, viewed October 20th 2020.

Liu, F. T., Ting, K. M., & Zhou, Z. H. (2008, December). Isolation forest. In 2008 Eighth IEEE International Conference on Data Mining (pp. 413-422). IEEE.

Sperotto, A., Schaffrath, G., Sadre, R., Morariu, C., Pras, A., & Stiller, B. (2010). An overview of IP flow-based intrusion detection. IEEE communications surveys & tutorials, 12(3), 343-356.

Xi, B. Adversarial machine learning for cybersecurity and computer vision: Current developments and challenges. Wiley Interdisciplinary Reviews: Computational Statistics, e1511.

Appendix 1 – Attackers and Victims for Task 1

Attackers (79)		
219.93.94.38	150.83.100.189	198.134.43.96
168.188.94.155	165.78.92.157	166.107.106.170
232.48.94.183	188.142.93.156	158.123.106.187
212.183.96.169	134.193.100.175	147.217.106.186
224.134.91.164	150.62.101.239	180.102.106.169
196.220.94.155	203.174.102.174	142.190.103.146
214.182.96.194	170.166.102.191	201.116.106.143
145.74.96.192	157.193.102.186	146.113.93.10
238.218.96.196	239.221.102.188	239.66.103.146
191.165.96.179	172.227.102.189	168.173.48.177
239.55.96.195	228.64.176.115	152.81.250.9
144.181.96.193	216.155.99.150	186.133.108.141
236.136.93.156	194.137.95.154	127.90.108.181
202.194.98.193	164.223.237.234	139.120.108.141
124.168.98.177	176.73.104.187	200.222.107.142
240.143.98.191	173.17.102.173	134.190.103.146
188.132.90.159	129.86.102.147	156.72.93.197
153.193.91.29	215.101.99.150	152.162.93.199
143.21.94.70	200.88.212.57	192.31.93.182
198.96.80.84	121.99.92.102	139.162.73.122
205.160.68.231	120.66.113.222	146.214.239.210
216.31.92.209	171.15.217.162	225.73.112.176
164.46.15.71	133.171.104.189	191.138.112.181
166.36.100.190	209.148.104.171	190.43.112.165
183.203.100.193	199.179.104.145	145.178.112.177
140.96.100.191	209.150.204.203	195.67.112.137
228.91.109.140		

Victims (101)		
229.173.94.6	169.120.91.100	139.105.91.7
127.57.106.59	159.142.91.136	214.220.116.49
237.22.91.151	201.160.91.137	151.239.126.16
180.167.91.139	140.159.94.20	127.103.97.14
133.22.92.81	183.179.91.4	198.103.91.184
229.114.92.105	132.186.91.146	125.197.92.108
154.18.91.101	123.69.91.204	216.134.121.157
236.101.93.85	135.232.91.135	184.220.172.106
221.34.91.52	202.18.114.51	219.211.91.93
189.63.94.9	235.100.94.3	124.29.91.21
125.189.87.2	194.178.91.33	169.17.118.47
220.206.99.23	238.128.91.30	208.233.91.239
160.108.108.57	230.151.91.29	228.90.27.6
209.131.113.52	139.168.91.68	190.189.94.47
195.231.110.55	134.236.91.67	174.62.91.4
148.205.90.101	183.137.91.10	137.168.98.3
141.143.97.78	157.114.94.38	171.139.94.222
195.154.92.141	134.18.91.18	146.225.93.150
148.78.116.13	185.31.91.5	182.176.128.89
239.145.91.120	237.73.94.2	146.204.94.31
172.163.91.179	184.141.93.4	131.77.94.207
201.157.199.6	206.100.124.115	207.110.123.87
205.224.112.53	124.26.91.3	168.111.126.232
232.19.91.2	128.29.91.2	221.35.94.4
227.36.92.3	220.158.91.144	153.234.91.72
145.51.93.53	142.46.94.16	190.236.92.77
190.59.93.68	201.71.91.13	145.116.120.45
230.49.118.110	201.209.93.27	141.111.103.62
220.174.133.87	217.145.94.97	191.42.91.205
196.192.125.2	145.41.117.86	126.106.93.110
154.195.93.62	123.166.91.131	175.225.124.41
171.206.94.137	176.221.130.153	154.78.92.159
220.50.118.9	183.24.91.182	182.141.104.3
157.124.91.198	167.84.93.4	

Appendix 2 – “.csv” and “.ipynb” files details

- “test_result_iforest_001_ipca.csv”: Our selected file as the best model for iForest.
- “test_result_iforest_001_allfeat.csv”: Secondary file for iForest.
- “test_result_iforest_001_var.csv”: Secondary file for iForest.
- “test_result_lof_001_ipca.csv”: Our selected file for analysis for LOF.
- “test_result_lof_001_allfeat.csv”: Secondary file for LOF.
- “test_result_lof_001_var.csv”: Secondary file for LOF.
- “task2_export.csv”: Required results for Task 2, includes adversarial samples.
- “task2_botnetIPs_export.csv”: Predicted Botnet IPs for Task 2.
- “iForest_all_001.ipynb”: iForest for 1% contamination.
- “iForest_all_002.ipynb”: iForest for 2% contamination.
- “iForest_all_01.ipynb”: iForest for 10% contamination.
- “iForest_all_025.ipynb”: iForest for 25% contamination.
- “iForest_all_data.ipynb”: iForest data processing.
- “iForest_results.ipynb”: iForest results.
- “LOF_all_0001.ipynb”: LOF for 0.1% contamination.
- “LOF_all_001.ipynb”: LOF for 1% contamination.
- “LOF _all_01.ipynb”: LOF for 10% contamination.
- “LOF_all_02.ipynb”: LOF for 20% contamination.
- “LOF_all_data.ipynb”: LOF data processing.
- “LOF_results.ipynb”: LOF results.
- “advSamplesTask2.ipynb”: Adversarial samples formulas and calculations.
- “featureGeneration.ipynb”: Feature generation functions.

Appendix 3 – Predicted botnet IPs for Task 2

Botnet IPs	
122.226.12.150	147.32.84.215
147.101.94.182	147.32.85.113
147.113.94.198	147.32.85.114
147.114.98.192	147.32.85.2
147.122.97.192	147.32.85.84
147.130.96.195	147.32.85.96
147.143.92.183	147.32.86.110
147.148.92.184	147.32.86.114
147.153.100.188	147.32.86.15
147.153.92.185	147.32.86.164
147.162.101.187	147.32.86.166
147.166.98.177	147.32.86.168
147.17.92.196	147.32.86.17
147.187.100.175	147.32.86.172
147.195.94.181	147.32.86.195
147.195.97.196	147.32.86.208
147.202.93.200	147.32.86.44
147.219.97.195	147.32.86.6
147.231.100.191	147.32.86.89
147.32.3.51	147.32.86.96
147.32.80.7	147.32.87.7
147.32.84.164	147.49.95.181
147.32.84.165	147.66.97.191
147.32.84.166	147.86.100.176
147.32.84.204	147.86.92.200
147.32.84.207	147.96.101.191
147.32.84.208	65.49.11.19
147.32.84.209	67.195.111.228
90.183.39.17	