Botnet Detection

1. Introduction

Connecting to internet is insecure and computers are exposed to various types of attacks, one of the most popular threats is the Botnet attack where the attacker takes control of a large number of machines in order to exploit some vulnerabilities in the security system, those machines are usually controlled by remote service that made available by some malware.

The attacker can do a lot of malicious acts such as Denial of Service attack (DoS), sending viruses and malwares, intruding, eavesdropping and so on, this risky problem encourage us to produce models to detect those intruders and prevent them from their bad intentions. One approach could be by using supervised learning techniques to produce a classifier model that detects the attackers, and that model will depend on a labeled dataset which produces another challenge, as the data in fields like this is private and in most cases the attacker is highly unknown.

In this document we are using a dataset provided that is a collaboration between Communications Security Establishment (CSE) and The Canadian Institute for Cybersecurity (CIC) and with help of AWS technologies, the dataset was produced using some tools that analyses the traffic from the raw data, specifically our data was collected in 03-02-2018 and it is available on UBN website.

2. Proposed approach

3. Experiments and results

Data Cleaning

Our techniques were implemented in google colab, we started by loading the data, taking a quick look into it, noticing that it is a large dataset containing around one million record and then we started data cleaning.

In data cleaning a check was performed for the null values and there were 2558 NULL values, and due to the size of the data we can just drop them. 5459 rows were duplicated so they were dropped, by seeing the summary statistics of the dataset, two features had infinite values, the feature of number of Bytes per second and the feature of number Packets per second.

Some columns had wrong type, so every flag column was converted into a categorical column, also the protocol column was converted into a categorical column as it has 3 unique protocols: 0, 6 and 17. However 10 features were redundant that is they have only one unique value (zero) so they were dropped.

Bar plot was used to determine the data balance and the percentage of botnets was 27.13% which is very acceptable for anomaly detection as in the figure 1. below:

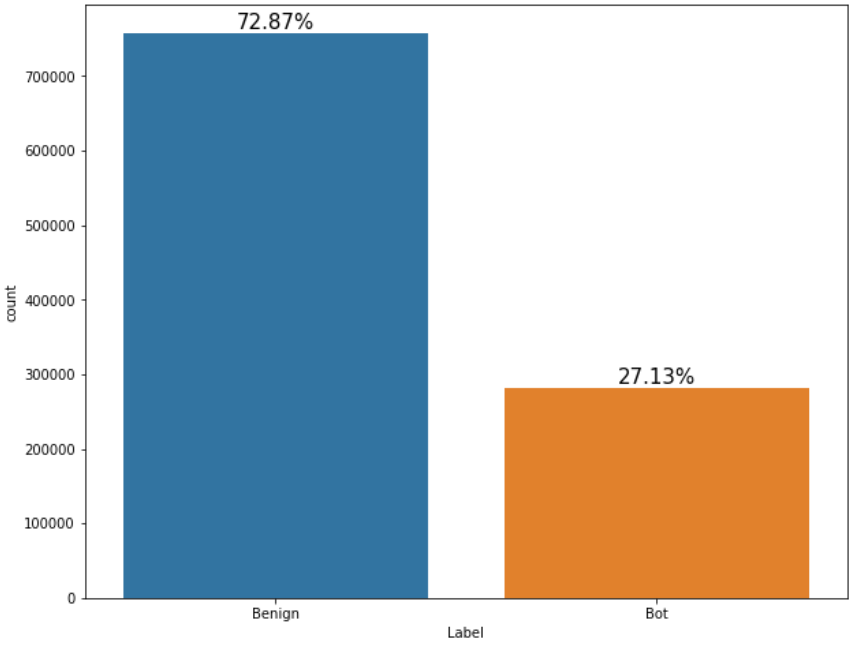

***Figure 1**. Checking Data Balance*

Because we have a large number of features, Principal Component Analysis (PCA) was conducted to reduce the dimensionality of the data into the most valuable features that captures 95% of the variation in the data, the analysis chose 19 principal components that explains around 95.7% of the variation in the data, so we started 80 features and end up with 19 Numerical features, 10 Categorical, 1 Timestamp and the outcome label.

As our data is collected in the same day we split the timestamp into date and time dropping date and Timestamp column, and formatting the time column, then in order to train the model we converted the labels into numbers, namely (0 for Benign and 1 for Bot), another thing is to convert the protocol column into a dummy variable as it has 3 unique values, and finally concatenate the principal components data frame with the rest of 14 other columns producing a cleaned data frame.

Training Models

After splitting data into training set with 70% of the data and testing set with 30% of the data, we trained some models as follows.

Decision Tree

For the training part, we started with Decision Tree as it is fast and robust algorithm, it could digest the whole one million record producing an interesting result with accuracy, precision, and recall all equals to

99.99%, at the first glance I thought that there is something wrong with my work but I then trusted those results after reading a part of this paper from this paper from springer journal of big data: “A survey and analysis of intrusion detection models based on CSE-CIC-IDS2018 Big Data”, which shows nearly the same results.

K-Nearest Neighbours

Due to resources limitations and for this model to work well we had to use a sample (20%) of the data, a model with k of 3 was trained on this sample and here the result was excellent with accuracy, precision, and recall equals 99.92%, 99.76% and 99.93% respectively.

Support Vector Machine

Here we even had to reduce the data into smaller sample (0.002%) that is around 2000 samples, and that reflected on the result, the accuracy, precision, and recall were 75.52%, 90% and 5.6% respectively.

Naïve Bayesian

For the naïve bayes classifier the sample was 20% percent of the data but the results was not good, accuracy, precision and recall were 52.10%, 36.11% and 98.38 respectively.

Neural Networks

For the neural networks model the sample was 20% of the data, and the Network had two hidden layers with 500 and 100 nodes respectively, here we divided the data into 70% for training, 15% for validation and 15% for testing, the training was on batch of size 64, we ran the model on 5 epochs and the accuracy was 99.31%, the learning plot is presented in Figure 2.

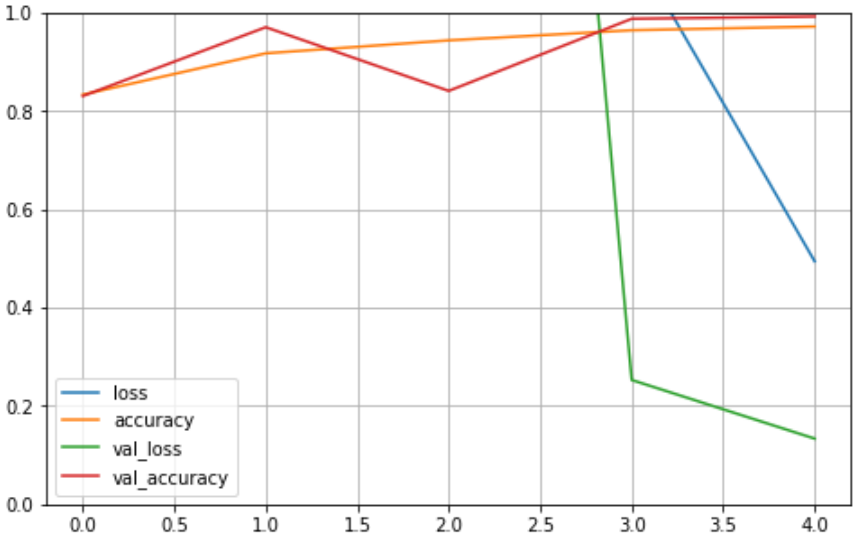

***Figure 2**. (5) Epochs*

Another try used 10 epochs but the accuracy was around 80% percent as in Figure 3.

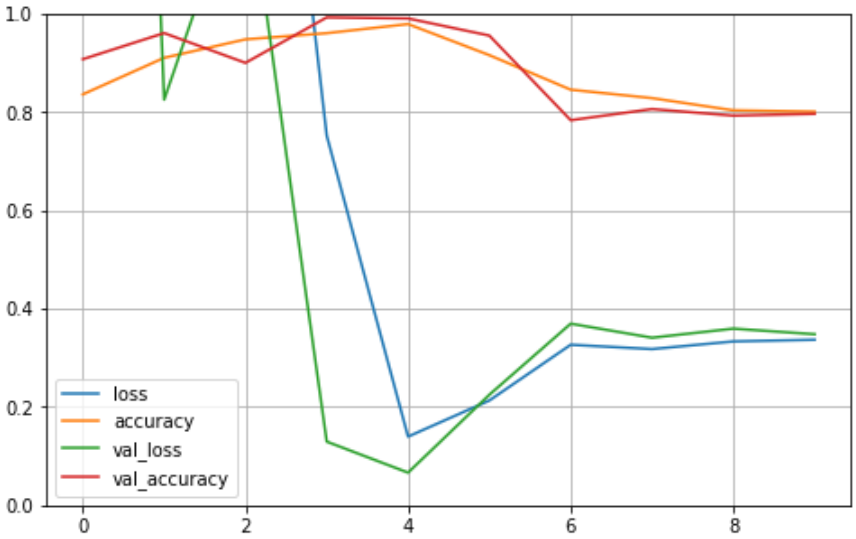

Figure 3. *(10) Epochs*

4. Conclusion

In conclusion, the best model was the Decision Tree model, and it could be used to help preventing attackers and intruders from violating the security system, but for further studies and with more powerful resources, I think that the research should be done on more than one day and with the utilization of GPUs to train good models on large scale datasets.

5. References

https://www.unb.ca/cic/datasets/ids-2018.html

Leevy, J.L., Khoshgoftaar, T.M. A survey and analysis of intrusion detection models based on CSE-CIC-IDS2018 Big Data. J Big Data 7, 104 (2020). https://doi.org/10.1186/s40537-020-00382-x

وَ الْـحَمْدُ للهِ رَبِّ الْـعَـالَـمِـيـنَ