Security Data Science - Project01

David Valenzuela, val171001@uvg.edu.gt
Carlos Chew, che17507@uvg.edu.gt

Abstract

The purpose of this project is to know the characteristics of a malware intrusion, implement machine learning models based on network traffic to determine possible intrusions, understand, and explain evaluation metrics for the selection of the best model, comprehend and apply techniques for the management of unbalanced information and promote research in data science applied to cybersecurity.

The project was developed with the python language and fed with the dataset of the SIMARGL project (SECURE INTELLIGENT METHODS FOR ADVANCED RECOGNITION OF MALWWARE AND STEGOMALWARE). The dataset has 6,570,058 observations of pure traffic, information that was useful for the elaboration of the models.

Introduction

The context of this research focused on learning new machine learning models with the aim of successfully classifying intrusions. The investigation was divided into eight parts to achieve detection: exploratory analysis, pre-processing, feature selection, data separation, implementation, evaluation metrics, ROC curve plots, and cross-evaluation with K-folds for k = 10.

The importance of this research revolves around teaching, in the same way, being able to implement these techniques in future projects, with the sole objective of making our future implementations more secure.

All research will be supported by the process that was carried out and the results obtained.

Theoretical framework

Random oversampling duplicates examples from the minority class in the training dataset and can result in overfitting for some models.

In statistics, an observation is simply one occurrence of something you're measuring.

Train/Test is a method to measure the accuracy of your model. It is called Train/Test because you split the data set into two sets: a training set and a testing set. 80% for training, and 20% for testing. You train the model using the training set.

Decision tree learning or induction of decision trees is one of the predictive modeling approaches used in statistics, data mining and machine learning. It uses a decision tree (as a predictive model) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves).

K-means clustering is a method of vector quantization, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster.

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned.

Methodology

Phase 1 - Preprocessing (coding, scales, use of techniques to solve the problem of the imbalance in the observations of the Slowloris attack).

In this phase, the pre-processing was carried out in which the value of the variables in the "LABEL" column was analyzed. Denial of Service Slowloris values are much less than the others. Oversample will be used once the observations are coded and separated into train and test.

Similarly, the relevant qualitative variables were coded.

The variables DST_TO_SRC_SECOND_BYTES, IPV4_DST_ADDR, IPV4_SRC_ADDR, SRC_TO_DST_SECOND_BYTES do not seem to have relevant information, we will not use them in the future.

Phase 2 - Feature Selection

Is the process of isolating the most consistent, non-redundant, and relevant features to use in the model construction. Scikit-learn API provides SelectKBest class for extracting best features of given dataset. The SelectKBest method selects the features according to the k highest score.

Phase 3 – Data Classification

Data was separated on this way:

- 55% train
- 15% validation
- 30% test

We focused on solving the problem of imbalance in the observations of the Slowloris attack.

We are going to use unsampled to 1816743 to have at least 50% of the feature that has the most observations.

Classification was supervised using machine learning approach, in which the algorithm learns from the data input provided to it — and then uses this learning to classify new observations.

Phase 4 – Implementation

The decision tree algorithm is implemented to train the model thanks to the test data, cross evaluation with K-folds for k = 10, validation data and the ROC curve plot.

The random forest algorithm is implemented to train the model thanks to the test data, cross evaluation with K-folds for k = 10, validation data and the ROC curve plot.

Results

Table #1 – Test data, decision tree

```
array([[1970452,    1848,      71,      21],
       [    950,  747767,       0,       0],
       [     17,       0,  682001,       0],
       [      1,       0,       0,  259234]])
```

Table #2 – Test data, decision tree

	precision	recall	f1-score	support
Normal flow	1.00	1.00	1.00	1972392
SYN Scan - aggressive	1.00	1.00	1.00	748717
Denial of Service R-U-Dead-Yet	1.00	1.00	1.00	682018
Denial of Service Slowloris	1.00	1.00	1.00	259235
accuracy			1.00	3662362
macro avg	1.00	1.00	1.00	3662362
weighted avg	1.00	1.00	1.00	3662362

Table #3 – Validation data, decision tree

```
array([[963203,    931,     33,     12],
       [   505, 367246,      0,      0],
       [     8,      0, 335603,      0],
       [     0,      0,      0, 127017]])
```

Table #4 – Validation data, decision tree

	precision	recall	f1-score	support
Normal flow	1.00	1.00	1.00	964179
SYN Scan - aggressive	1.00	1.00	1.00	367751
Denial of Service R-U-Dead-Yet	1.00	1.00	1.00	335611
Denial of Service Slowloris	1.00	1.00	1.00	127017
accuracy			1.00	1794558
macro avg	1.00	1.00	1.00	1794558
weighted avg	1.00	1.00	1.00	1794558

Table #5 – ROC, decision tree

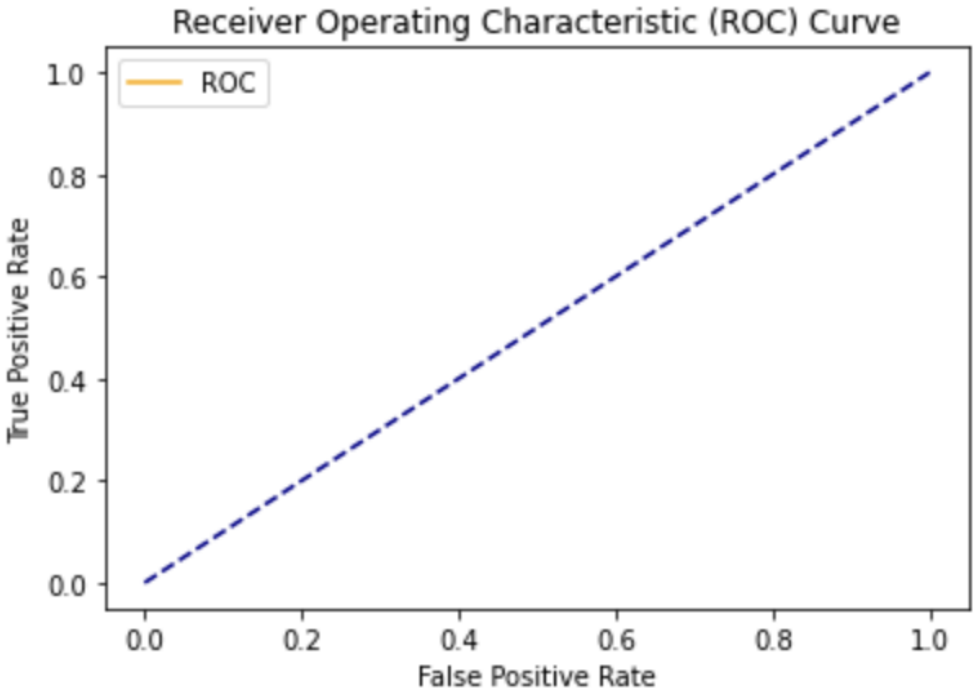

Table #6 – ROC, decision tree

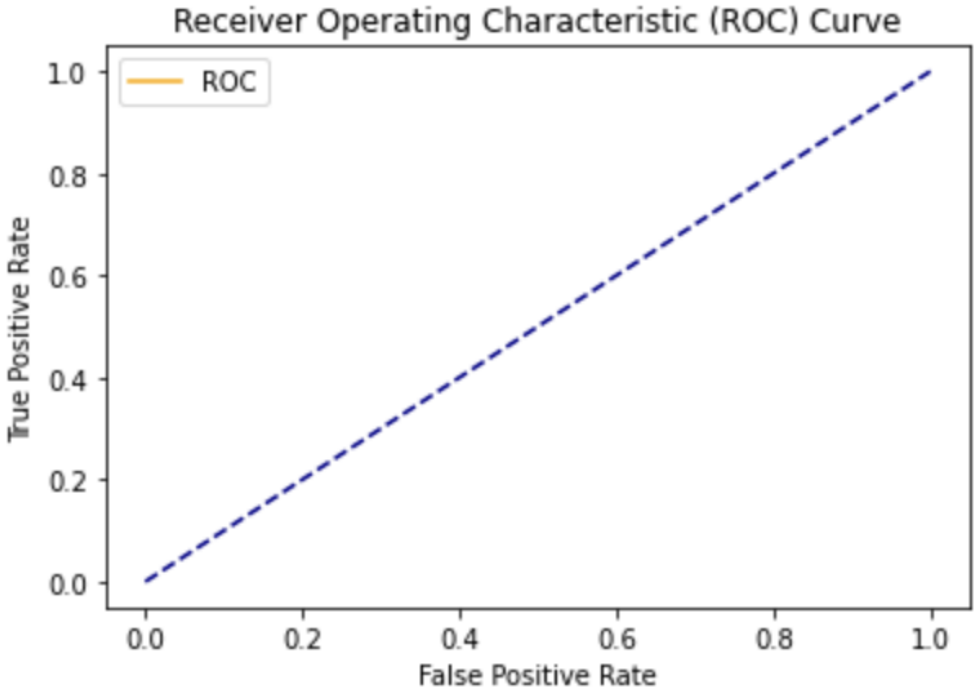

Table #7 – K-folds, decision tree

```
array([0.9990143 , 0.99901703, 0.99883409, 0.9989433 , 0.9990416 ,
       0.9990416 , 0.99898699, 0.99899791, 0.99901703, 0.99895696])
```

Table #8 – K-folds, decision tree

```
array([0.99893567, 0.99903598, 0.9989301 , 0.99891338, 0.99904155,
       0.99889109, 0.99888552, 0.99889667, 0.99899696, 0.99888551])
```

Table #9 – Test data, Random Forest

```
array([[ 209405,  104889, 1203044,  455054],
       [     82,  748609,      26,       0],
       [      0,       0,  682018,       0],
       [      0,       0,      16,  259219]])
```

Table #10 – Test data, Random Forest

	precision	recall	f1-score	support
Normal flow	1.00	0.11	0.19	1972392
SYN Scan - aggressive	0.88	1.00	0.93	748717
Denial of Service R-U-Dead-Yet	0.36	1.00	0.53	682018
Denial of Service Slowloris	0.36	1.00	0.53	259235
accuracy			0.52	3662362
macro avg	0.65	0.78	0.55	3662362
weighted avg	0.81	0.52	0.43	3662362

Table #11 – Validation data, Random Forest

```
array([[102704,  51141, 587932, 222402],
       [    42, 367696,     13,      0],
       [     0,      0, 335611,      0],
       [     0,      0,      4, 127013]])
```

Table #12 – Validation data, Random Forest

	precision	recall	f1-score	support
Normal flow	1.00	0.11	0.19	964179
SYN Scan - aggressive	0.88	1.00	0.93	367751
Denial of Service R-U-Dead-Yet	0.36	1.00	0.53	335611
Denial of Service Slowloris	0.36	1.00	0.53	127017
accuracy			0.52	1794558
macro avg	0.65	0.78	0.55	1794558
weighted avg	0.81	0.52	0.43	1794558

Table #12 – ROC, Random Forest

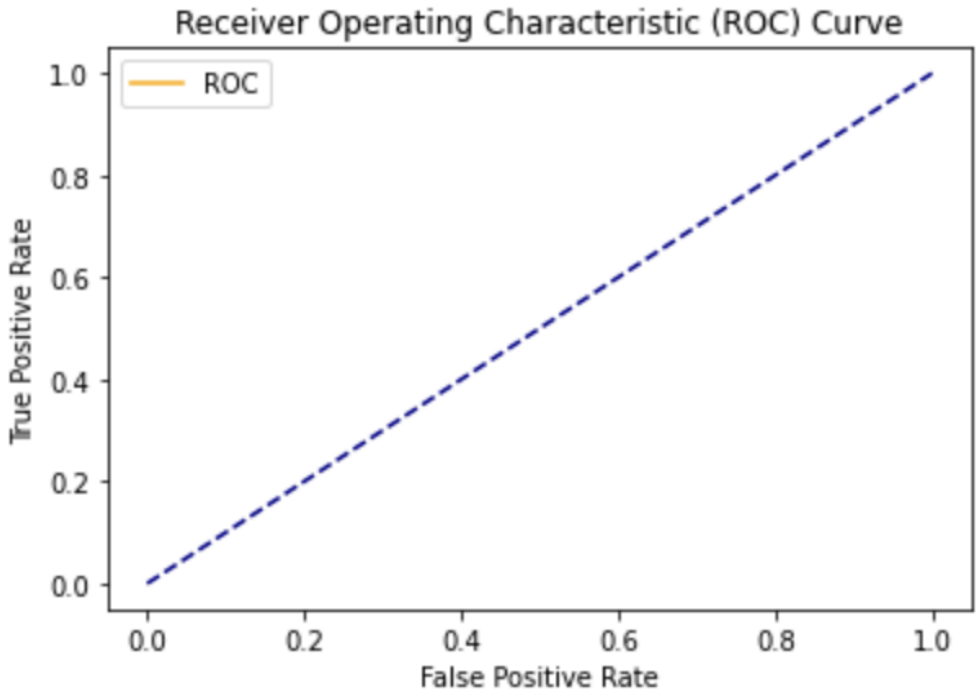

Table #13 – ROC, Random Forest

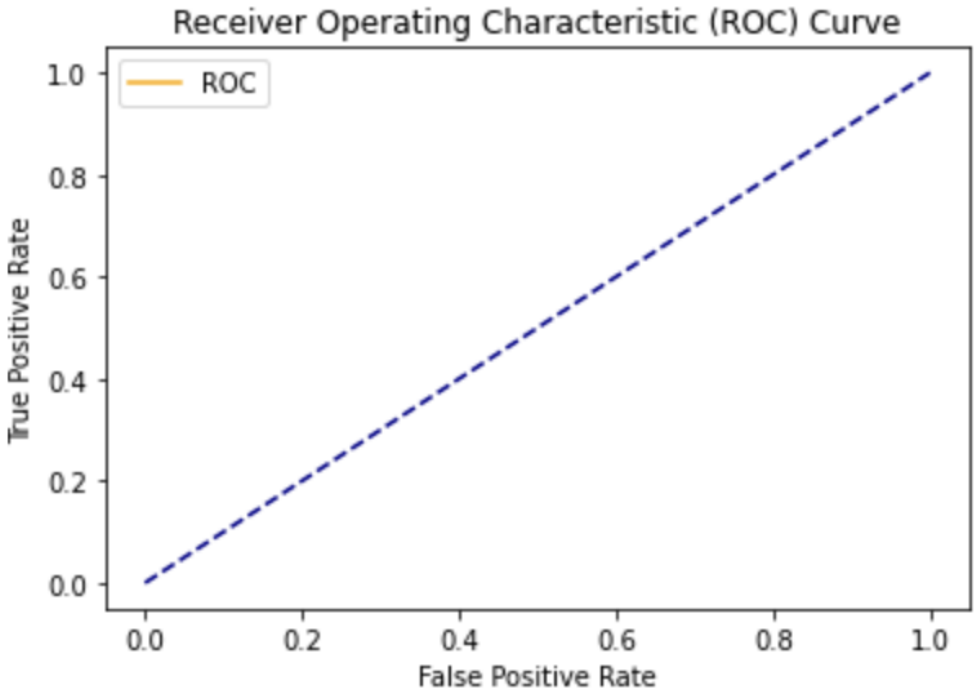

Table #14 – k-folds, Random Forest

```
array([0.83906596, 0.83847345, 0.83816173, 0.83870783, 0.83858769,
       0.83875152, 0.83923481, 0.83922116, 0.83821361, 0.8395297 ])
```

Table #15 – k-folds, Random Forest

```
array([0.83935338, 0.83978803, 0.83803829, 0.83947597, 0.8383392 ,
       0.83792127, 0.83792127, 0.8395707 , 0.83846647, 0.83931348])
```

Conclusions

1. Random Forest became an incredible calculation, for both classification and relapse issues, to create a prescient demonstrate. Its default hyperparameters as of now return incredible comes about and the framework is awesome at dodging overfitting. In this attack detection research, it worked.
2. Decision trees are capable visualization algorithm that help decision-makers to form the correct moves at the correct time. The tree creates a visual representation of all possible outcomes, rewards, and follow-up decisions in one document.

3. It was possible to know the characteristics of a malware intrusion, two machine learning models were implemented based on network traffic to determine possible intruders and research in data science applied to cybersecurity was promoted.

References

Brownlee, J. (2021, 5 enero). *Random Oversampling and Undersampling for Imbalanced Classification*. Machine Learning Mastery. https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/

Python Machine Learning Train/Test. (2022). Train and Test. https://www.w3schools.com/python/python_ml_train_test.asp

Wikipedia contributors. (2021, 31 diciembre). *Decision tree learning*. Wikipedia. https://en.wikipedia.org/wiki/Decision_tree_learning

Wikipedia contributors. (2022a, febrero 10). *Random forest*. Wikipedia. https://en.wikipedia.org/wiki/Random_forest

Wikipedia contributors. (2022b, febrero 11). *K-means clustering*. K-Means. https://en.wikipedia.org/wiki/K-means_clustering