
Universita degli studi di Salerno

Department of Computer Science

# Malware Prediction through ML models

June 8, 2022

Project Report

Fatima (f.baloch@studenti.unisa.it)

Waseem Afzal (w.afzal@@studenti.unisa.it)

Supervised by:
Prof. Fabio Palomba & Dario Di Nucci
Corresponding Tutors
Giulia Sellitto & Giammaria Giordano

# Contents

## Abstract

In this project, we used the CTU-13 datasets to train Random Forest clas- sifier models capable of classifying IP addresses that belong to a botnet. A Random Forest Classifier (Scikit-Learn package) was the best model for cor- rectly classifying malicious IP addresses in the test dataset, with an accuracy of 95source bytes, and total bytes of a bidirectional net flow were the most essential elements in categorizing malicious IP addresses using feature importance, per- mutation importance, and SHAP model explanation approaches. Malicious gamers used UDP (User Datagram Protocol) the most of any protocol, with traffic originating from UDP boosting the possibility that a specific net flow belongs to them.


## 1 Introduction

CTU University in the Czech Republic captured the CTU-13 dataset in 2011. Its goal was to create a dataset with botnet traffic, regular traffic, and back- ground traffic, which resulted in 13 sets that were pooled for most of the anal- ysis. CTU University used unidirectional net flows for model training in an empirical evaluation of botnet detection approaches. However, subsequent training revealed that bidirectional net flows produced better results. This project used bidirectional net flows as a result of that precedent. According to the CTU University classification method, the aggregated dataset made up of all 13 datasets comprised 20,643,076 flows, with 432,755 botnet flows, 369,806 regular flows, and 19,840,515 miscellaneous flows (mainly background flows). The data had to be balanced to produce more accurate predictions without ex- cessive noise. The goal feature was whether an IP address belonged to a botnet (bots included Neris, Rbot, Virut, Menti, Soguo, Murlo, and NSIS.ay). This balancing procedure can be found in the data preparation section.

## 2 Goal Definition

This project aims to develop a model that can accurately classify malicious activity in a bidirectional network flow with at least 90% accuracy.

## 3 DATA UNDERSTANDING

### 3.1 Data Collection Overview

CTU University previously collected the data; thus, gathering it was simple. Each of the 13 datasets was turned into a single pandas data frame after the data was loaded into Google Colab (a Jupyter equivalent). The information was then merged into a single larger data frame.

### 3.2 Data Description

StartTime, Dur, Proto, SrcAddr, Sport, Dir, DstAddr, Dport, State, sTos, dTos, TotPkts, TotBytes, SrcBytes, and Label are among the features included in each of the 13 labeled net flow files. Only Dur (Duration), TotPkts (Total Packets), SrcBytes (Source Bytes), and the dummy features of the Proto feature (Protocol) were used for this project. Furthermore, the target feature "Malicious" was created using the CTU University list of known malicious IP addresses. According to the CTU University classification method, the aggregated dataset made up of all 13 scenarios comprised 20,643,076 flows, with 432,755 botnet flows, 369,806 regular flows, and 19,840,515 miscellaneous flows (mainly background flows).

### 3.3 Data Exploration

The data was examined by counting the number of rows available and, in the case of particular features, printing the unique values to check if one hot encoding of the feature was feasible and valuable. The data types of each feature were also noted, though features used in Gaussian Naïve Bayes models should be normalized to floating in future project replications to minimize errors. This can be disregarded even if one does not plan to use Naïve Bayes.

### 3.4 Data Quality Check

The data quality was maintained because most of the data set was cleansed in advance by CTU University (e.g., NA values). Furthermore, because the data was solely made up of converted PCAP file logs, there was a minimal chance for error unless conversion to CSV and pandas data frame format issues occurred, which were not seen.

## 4 Data Preparation

### 4.1 Data Selection

Initial models were generated on a per-scenario basis to be better acquainted with the data and avoid potential pitfalls when working with aggregated data. After random forest classifiers were generated for each of the 13 scenarios, it was best to aggregate all 13 scenarios into one data frame before utilizing an 80/20 train/test split to ensure that the trained models were not overfitting and becoming acquainted with certain features of a singular bot.

### 4.2 Data Cleaning

As previously indicated, the data did not require extensive cleaning because it was already a processed dataset. However, as indicated in section 3.2 – Data Description, superfluous elements were removed from the data frame to conserve storage and computing power. Reducing extraneous features was a natural choice because the data frame had so many records, especially given Google Drive's storage restrictions and Google Colab's GPU limitations. The malicious target column was constructed before the extraneous features were removed, and the identifying column SrcAddr was removed to eliminate any potential source of bias. Furthermore, because malicious net flows made up only about 2.1 percent of the whole data set, the data was scaled so that the number of non-malicious flows matched the number of malicious flows. This was done because a model trained on a highly skewed data set is especially vulnerable to false predictions due to excessive noise. To verify that there was no selection bias. So, the non-malicious flows were assigned at random. As we will see later, balancing the data resulted in significantly higher accuracy, f1 scores, and ROC AUC values.

## 5 Data Modeling

### 5.1 Technique Selection

We planned to utilize a range of alternative models in this project, such as support vector machines, random forest, and Gaussian Naïve Bayes. However, the results produced by the initial random forest classifier model made further research unnecessary. If the experiment were extended further, more models would be used to observe the complete range of findings and get the best final results.

## 5.2 Data Splitting

Even after the data set had been balanced, an 80 percent /20 percent train test split was considered sufficient. The final data set contained around 900,000 records, and split tuning had only a minor impact in early tests. If this project were to be reproduced, k folds validation would likely be used again, but the findings were promising and showed no signs of overfitting. Therefore 80/20 was accepted.

## 5.3 Model Generation

The original goal of this project was to create four or five different models (RFC, Naïve Bayes, SVM, and so on), compare them, and choose the best one. Minor tweaking of the random forest classifier, on the other hand, resulted in highly accurate models, making further investigation redundant save as a possible portfolio exercise. An initial Naive Bayes model was created. However, the results were significantly worse than the baseline random forest model. Therefore hyperparameter tuning was considered superfluous in the absence of a robust alternative model.

### 5.3.1 Per-Datasets Classifiers

13 random forest classifier models were created at first, one for each dataset. These models were created as a starting point and were not fine-tuned in any way to better grasp the data and identify potential issues. On the other hand, the goal feature was still unbalanced, and the model metrics had not yet been determined. As a result, these models were useless as a finished product. The benefit of these models was that they revealed the necessity to balance the target feature, and the area under the ROC curve was included in the final model's evaluation.

### 5.3.2 Random Forest Classifier

As discussed in 6a – Result Evaluation, the random forest classifier model created was deemed to be the best model. With the maximum leaf nodes set to 5, 50, 500, and 5,000, two functions were created to assess the area under the ROC curve and the accuracy score of a baseline model. The 500 maximum leaf nodes parameter was chosen since it produced the highest metric values with the least amount of runtime. With 5000 maximum leaf nodes, there were marginal benefits, but training time was substantially longer, and the model was more likely to be overfit. Due to the vast size of this dataset, Randomized Search CV and Grid Search CV were explored for further hyperparameter optimization. However, they were deemed unnecessary because the original metrics were regarded as acceptable for the aims of this research. Grid searching might be used to optimize hyperparameters such as minimum impurity split, minimum samples per leaf, verboseness, max depth, max features, and several other parameters if runtime was not a concern.

### 5.3.3 Naïve Bayes

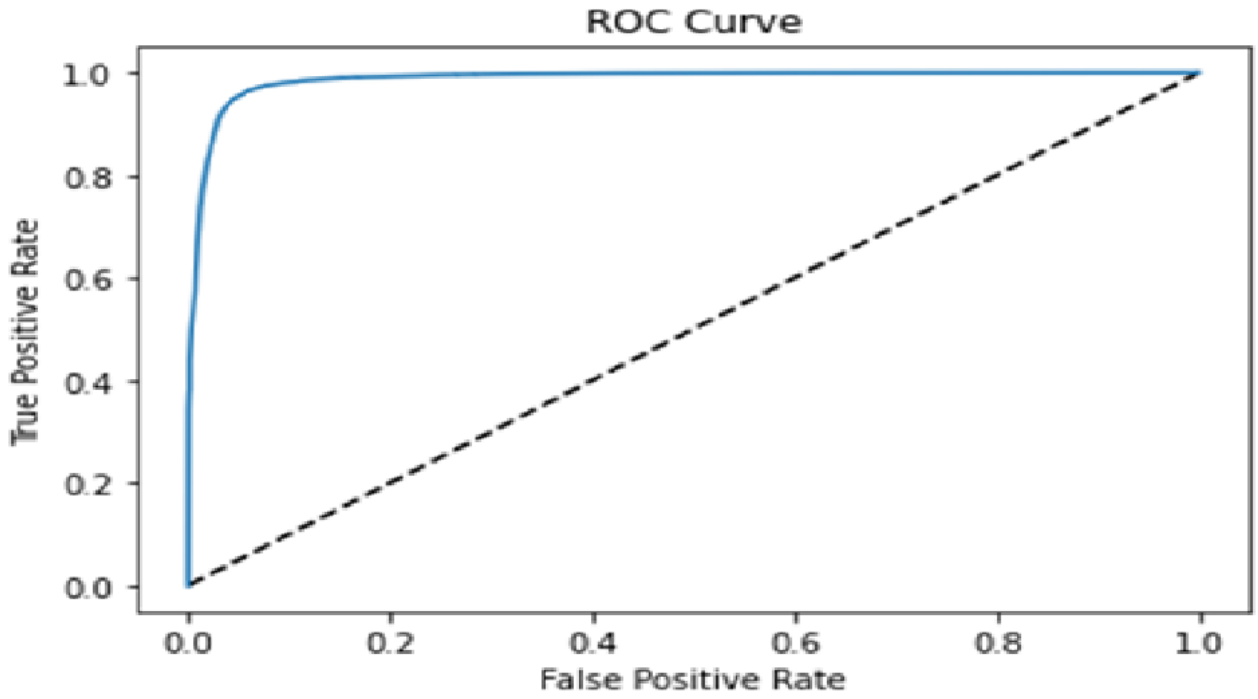
An early Naive Bayes model was created to investigate if a different model may outperform the random forest classifier model. The basic Gaussian Naïve Bayes model, on the other hand, performed at a considerably lower level than the random forest. Therefore, further tuning was avoided. Model Assessment's Gaussian Naïve Bayes section delves more into the logic behind this decision.

## 5.4 Model Assessment

Confusion matrices, accuracy scores, ROC AUC ratings, and f1 values evaluated the models. These metrics were chosen because a high ROC AUC emphasizes the actual positive rate, a higher accuracy score puts much

emphasis on accurate predictions in general, and f1 is the harmonic mean of the model's accuracy and recall. Hence, a model with high values in each of these metrics performs well regardless of one's philosophy on the most essential (True positive rate, false-positive rate, actual negative rate, false-negative rate).

The actual positive rate would be the most crucial indicator if the chosen random forest model were not very precise. False positives may be forgiven more readily in this instance because IP addresses labeled as malware could later be "forgiven," however false negatives would be penalized more severely due to the lack of bot detection in the network.




### 5.4.1 Random Forest Classifier

With a ROC AUC of 0.949, an f1 score of 0.949, and an accuracy of 0.949, the random forest classifier model delivers highly accurate predictions and correctly classifies malicious IP addresses within the test data with an acceptable error rate. The model predicted 85,171 true negatives and 83,752 true positives out of 177,855 test records. Alternatively, it produced 3,576 erroneous optimistic predictions (2 percent of the test set) and 5,356 false-negative predictions (3 percent of the test set), indicating that the model was 95 percent correct. To understand which features most strongly impacted the model's prediction, we utilized Shapley Additive Explanations (SHAP), feature importance, and permutation importance to create a complete picture of the model's inner workings (exploring the "black box"). Shapley Additive Explanations (SHAP), feature importance, and permutation importance were used to generate a detailed image of the model's inner workings (exploring the "black box") in order to figure out which features had the most significant impact on the model's prediction.

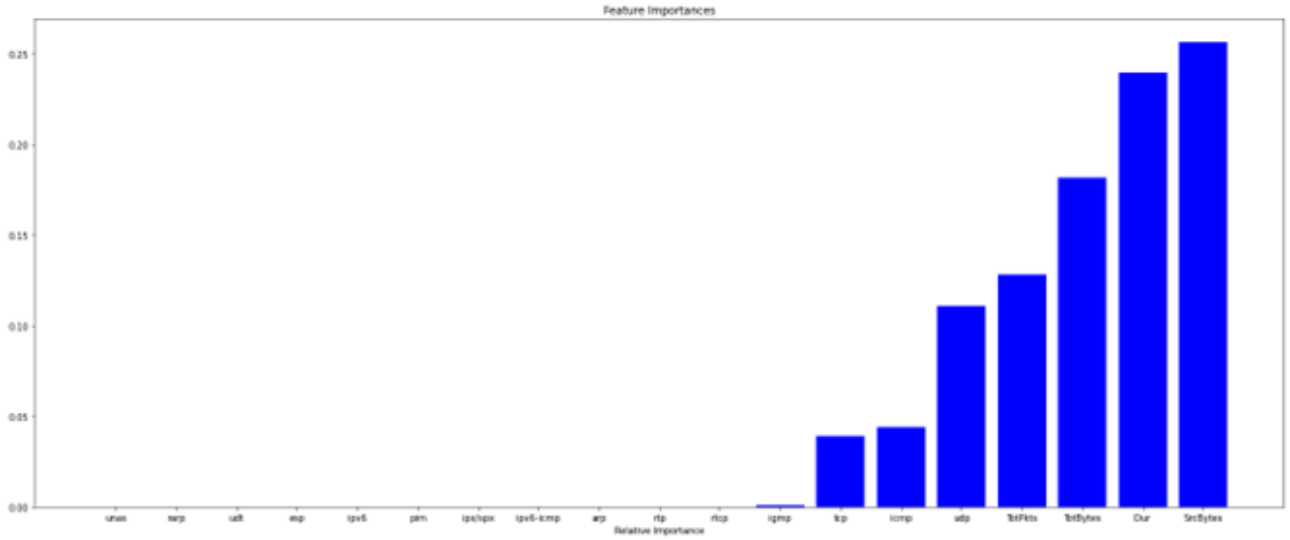



**Figure 1:** Feature Importance Graph for RFC Model

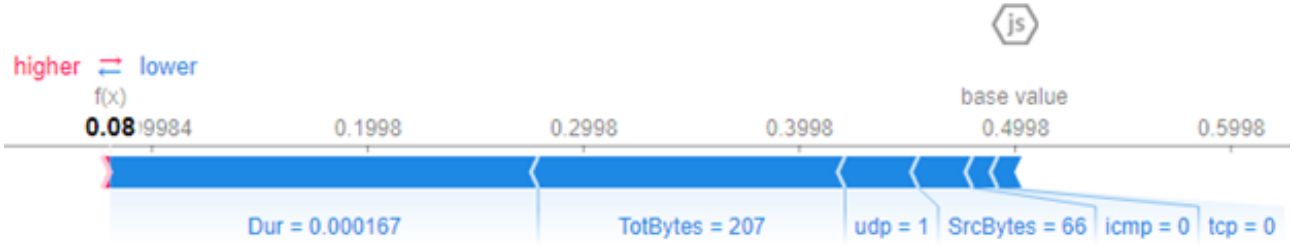
The random forest's built-in feature importance function was used to calculate feature importance in this graph, showing which features significantly impacted the model's predictions. While it is a less complex assessment of the most significant characteristics, it is a reliable estimate. Duration, Source Bytes, Total Bytes, Total Packets, UDP, ICMP, and TCP are the most crucial criteria in identifying whether a flow is from a malicious source, as shown in the first diagram. The majority of the other protocols using this metric had no impact on the model's predictions. While this corresponds to other Blackbox descriptions rather well, more accurate models are still to come.

| Weight | Feature |
|---|---|
| 0.3535 ± 0.0012 | SrcBytes |
| 0.2242 ± 0.0022 | Dur |
| 0.1941 ± 0.0011 | TotBytes |
| 0.1513 ± 0.0009 | udp |
| 0.1376 ± 0.0006 | icmp |
| 0.1122 ± 0.0010 | tcp |
| 0.1009 ± 0.0010 | TotPkts |
| 0.0000 ± 0.0000 | rtcp |
| 0.0000 ± 0.0000 | arp |
| 0.0000 ± 0.0000 | igmp |
| 0 ± 0.0000 | esp |
| 0 ± 0.0000 | unas |
| 0 ± 0.0000 | ipv6 |
| 0 ± 0.0000 | udt |
| 0 ± 0.0000 | ipx/spx |
| 0 ± 0.0000 | pim |
| 0 ± 0.0000 | rarp |
| 0 ± 0.0000 | ipv6-icmp |
| -0.0000 ± 0.0000 | rtp |

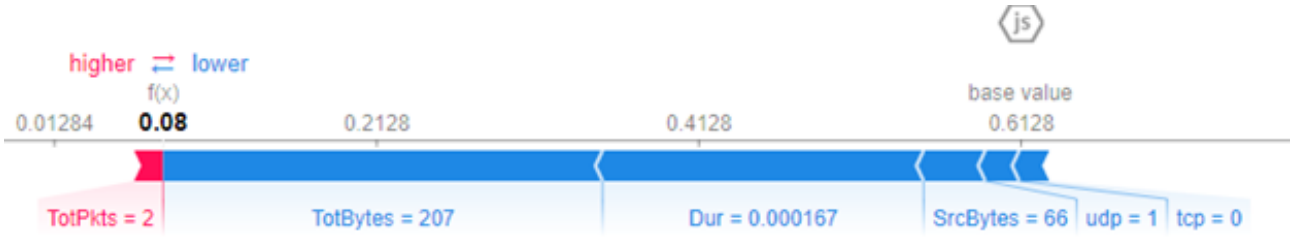
**Figure 2:** Permutation Importance Graph for RFC Model

The permutation significance values for the significant classifier model are shown in the diagram above. For each iteration, permutation significance works by shuffling the values within a particular feature while keeping all other features constant. Shuffling a feature should theoretically affect the model's prediction accuracy, and the difference in accuracy was assessed in permutation significance values. In general, shuffled more essential predictive elements had the most significant impact, so Source Bytes, Duration, Total Bytes, UDP, ICMP, Total Packets, and TCP were the features that had the most significant impact. Furthermore,

because alternative protocols other than UDP, ICMP, and TCP make up a lower share of observed flows, it seems to sense that they are assigned less weight in this paradigm.




SHAP values interpret the impact of having a certain value for a given feature in comparison to the prediction that would be made if that feature took some baseline value. This visual shows that TotPkts, TotBytes, Dur, SrcBytes, and udp have the greatest impact on the predictions (this lines up very nicely with our permutation importance). TotPkts being equal to 2 and TotBytes being equal to 207 increase the prediction value, and the largest impacts come from Dur being 0.000167 and SrcBytes being 66.




This is an approximation of the previous plot using kernel explainer. Kernel SHAP is a method that uses a special weighted linear regression to compute the importance of each feature. Less accurate, but roughly the same story.

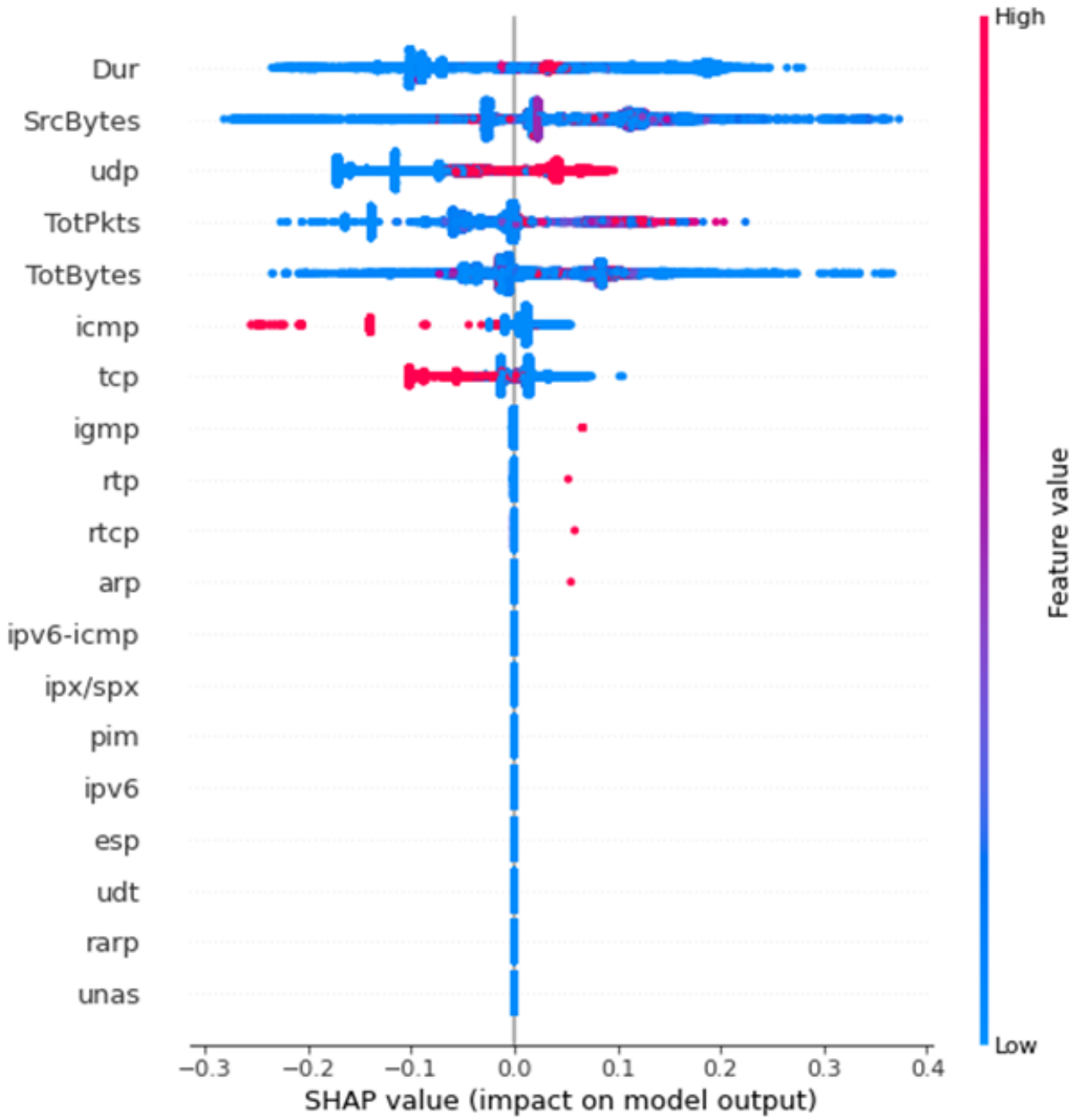



**Figure 3:** SHAP Summary Plot for RFC Model

The SHAP summary charts show which points have a positive or negative impact on the model's output and to what extent. The horizontal position indicates how much a certain point influences the result, while the color indicates whether the value was high or low for that particular row of data. The hue is skewed since the bulk of the features in this data set were binary. Furthermore, SHAP summary graphs are inefficient when working with massive data sets. As a result, this SHAP plot was created using only a quarter of the test data (25%) and may not accurately reflect the overall trend of the data. Regardless, it should be considered a reliable estimate.

This plot is pretty consistent with the rest of the model exploration features, indicating that Duration, Source Bytes, UDP, Total Bytes, Total Packets, ICMP, and TCP have the most significant effects on the model's output, with Duration and Source Bytes being the two most important features across the board. Because of the coloring difficulties, it is challenging to identify how these particular elements affect the model. For example, duration is practically blue, indicating that while most of those values were considered modest for their rows, they significantly impacted the model in both directions. As such, fewer insights were derived

here than those that would have been derived within another dataset, but the insights gained are valuable. This model confirms that TCP and ICMP increased the chance that a given flow is malicious, which may not have been apparent given only feature and permutation importance.

#### 5.4.2 Gaussian Naïve Bayes

The Gaussian Naïve Bayes model had a ROC AUC of 0.543, an f1 score of 0.681, and an accuracy score of 0.542. Although more hyperparameter tuning could have improved these metrics, the high metrics observed in the random forest classifier model made this unnecessary for the study's goals.

This model adequately predicted 86,761 malicious flows and 9,923 non-malicious flows compared to the Random Forest Classifier model. It also produced 78,824 false negatives and 2,347 false positives, ultimately rendering the model unusable. Of fact, more hyperparameter tuning could have enhanced this outcome, but given the random forest model's overwhelming performance, it was unnecessary. The Gaussian Naïve Bayes model was not selected as the final model, and its weak metrics rendered it unnecessary to investigate the black box.

## 6 Evaluation

### 6.1 Result Evaluation

Overall, the experiment was a success, with a model that was 95 percent accurate. Minor improvements might be made by focusing less on runtime and fine-tuning hyperparameters, but this model is now enough.

### 6.2 Process Review

If the knowledge that we had now when the project started, this process would have gone much faster. Initial models were discarded because they used unidirectional flows. Suppose the final model had not been a simple adjustment of the baseline random forest classifier with a balanced training dataset. In that case, this project could have taken longer and cost a lot more money. More thorough planning will be required to prevent such issues if an appropriate model is not immediately evident. Beyond the immediate success of the final model, this project existed as a learning exercise, and it was a massive success in that aspect.

### 6.3 Deployment Preparation

While the finished model will not be integrated into a regular workflow, efforts to prepare it for future use could be performed. Pipelines might be created using Pyspark or a comparable library to automate the data cleaning process, including imputing and feature extraction activities. Cleaning and model development were done by hand for this project, but pipelines would be required for large-scale deployments to ensure a consistent procedure that did not use too much time and resources. The model training may be turned into a pipeline or function for true automation and replicability with datasets other than the CTU-13 data.

## 7 Project Review

This project simply created a classification model capable of accurately categorizing malicious net traffic while allowing for some false positives. The project was a success in this regard.

## References

[1] S. García, M. Grill, J. Stiborek, A. Zunino, An empirical comparison of botnet detection methods, Computers Security, https://doi.org/10.1016/j.cose.2014.05.011. (https://www.sciencedirect.com/science/article/pii/S0167404814000923)

[2] Garcia, Sebastian. Malware Capture Facility Project. Retrieved from https://stratosphereips.org

[3] Haghighat, Mohammad Hashem Li, Jun. (2018). Edmund: Entropy based attack Detection and Mitigation engine Using Netflow Data. 1-6. 10.1145/3290480.3290484.

[4] D. C. Le, A. Nur Zincir-Heywood and M. I. Heywood, "Data analytics on network traffic flows for botnet behaviour detection," 2016 IEEE Symposium Series on Computational Intelligence (SSCI), 2016, pp. 1-7, doi: 10.1109/SSCI.2016.7850078.

[5] Vishwakarma, A. R. (n.d.). Real-time ad click fraud detection - SJSU scholarworks. scholarworks. Retrieved May 30, 2022, from https://scholarworks.sjsu.edu/cgi/viewcontent.cgi?article=1916context=etd_projects

**GitHub Repository:** https://github.com/waseemafzal70/Malware-Predection-Software-Dependability.git
**Overleaf:** https://www.overleaf.com/read/kdwyzxgjmnrb