CTU-13 Botnet Classification

Neil Andrew Collins
Leeds School of Business
University of Colorado, Boulder
Boulder, CO, USA
neil.collins@colorado.edu

***Abstract*—This project trained Random Forest and Naïve Bayes classifier models on the CTU-13 dataset's bidirectional net flows to create models capable of classifying IP addresses that belong to a bot net. The optimal model was a Random Forest Classifier (Scikit-Learn package) that correctly classifies malicious IP addresses within the test dataset with an accuracy of 0.950, an F1 of 0.949, and an area under the ROC curve of 0.950. Utilization of feature importance, permutation importance, and SHAP model explanation methods found that the most important features in classifying malicious IP addresses were the duration, source bytes, and total bytes of a bidirectional net flow. Of all protocols, malicious players most commonly utilized UDP (User Datagram Protocol), with traffic being sourced from UDP increasing the likelihood that a certain net flow belongs to a bot IP address by approximately 15%.**

Keywords—Botnet, Machine Learning, Classification, CTU-13, Scikit-Learn, SHAP, feature importance, permutation importance

I. INTRODUCTION

The CTU-13 dataset was captured by CTU University in the Czech Republic in 2011. It aimed to produce a dataset made up of botnet traffic, normal traffic, and background traffic, resulting in 13 scenarios that were combined together for the bulk of this analysis. In *An empirical comparison of botnet detection methods*, CTU University utilized unidirectional net flows for their model training, though further training found that bidirectional net flows led to better outcomes. Based on that precedent, this project utilized bidirectional net flows.

The aggregated dataset made up of all 13 scenarios totaled 20,643,076 flows, consisting of 432,755 botnet flows, 369,806 normal flows, and 19,840,515 other flows (primarily background flows) per the CTU University labeling process. Since the target feature was whether or not an IP address belonged to a botnet (bots included Neris, Rbot, Virut, Menti, Soguo, Murlo, and NSIS.ay), the data had to be balanced to get more accurate predictions without excessive noise. The process for this balancing can be found under the data preparation section.

II. BUSINESS UNDERSTANDING

A. Business Objective Outline

The objective of this project is to generate a relatively accurate classifier model that is able to detect the presence of malicious IP addresses within the bidirectional net flows of a business. While this model may be limited in the specific bots that it can detect, an improvement on the baseline is expected. Bots can cause a variety of issues ranging from interfering with web traffic analytics to hosting malicious software, so having a better way to identify the traffic of malicious IP addresses is beneficial in multiple ways.

B. *Situation Assessment*

Though this is a toy model generated from controlled data, it sets a valuable framework for future analysis regarding botnet activity in other situations. Malicious bots are a major threat in the realm of cybersecurity, and we must constantly take measures to counteract the threat that they pose.

C. *Goal Definition*

The goal of this project is to create a model that can classify malicious activity in a bidirectional net flow with at least 90% accuracy.

III. DATA UNDERSTANDING

A. *Data Collection Overview*

Data collection was relatively straightforward, as the data was already collected by CTU University. The data was imported into Google Colab (Jupyter alternative), with each of the 13 scenarios being made into a single pandas data frame. Afterward, the data was combined into a single larger data frame.

B. *Data Description*

The data exists in 13 labeled net flow files, each including an index column as well as the features StartTime, Dur, Proto, SrcAddr, Sport, Dir, DstAddr, Dport, State, sTos, dTos, TotPkts, TotBytes, SrcBytes, and Label. For the needs of this project, only Dur (Duration), TotPkts (Total Packets), SrcBytes (Source Bytes), and the dummy features of the Proto feature (Protocol) were utilized. Furthermore, the target feature "Malicious" was generated based on the known malicious IP addresses provided by CTU University.

As previously stated, the aggregated dataset made up of all 13 scenarios totaled 20,643,076 flows, consisting of 432,755

botnet flows, 369,806 normal flows, and 19,840,515 other flows (primarily background flows) per the CTU University labeling process.

C. *Data Exploration*

The data was explored via the observation of how many rows were available, and in the case of certain features the unique values were printed to see whether or not one hot encoding of the feature was plausible and/or beneficial. The data types of each feature were observed as well, though in future project replications features utilized in Gaussian Naïve Bayes models should be normalized to floats to avoid errors. If one does not intend to utilize Naïve Bayes, this can be ignored.

D. *Data Quality Check*

The majority of the data set was cleaned in advance by CTU University (e.g. NA values), and as such the data quality was upheld. Furthermore, as the data was entirely made up of converted PCAP file logs, there was little room for error to occur unless errors occurred in conversion to CSV and to pandas data frame format, though no such errors were observed.

IV. Data Preparation

A. *Data Selection*

Initial models were generated on a per-scenario basis as a way to better become acquainted with the data and to avoid any potential pitfalls that may arise when working with aggregated data. After random forest classifiers were generated for each of the 13 scenarios, it was seen as 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 being overfit and becoming acquainted with certain features of a singular bot.

B. *Data Cleaning*

As previously stated, the data did not require a vast amount of cleaning due to it already being a processed dataset. However, unnecessary features were dropped from the data frame to save on storage and computing power as described in section 3b – Data Description. The data frame included so many records that removing unnecessary features was natural considering the limitations of storage in Google Drive and the limitations of GPU in Google Colab. Before the unnecessary features were dropped, the malicious target column was generated and the identifying column SrcAddr was dropped to remove any potential source of bias.

Furthermore, since malicious net flows only consisted of approximately 2.1% of the entire data set, the data was scaled so that the number of non-malicious flows were equal to the number of malicious flows. This was done because a model trained on an overly skewed data set is highly prone to inaccurate predictions due to excessive noise. The non-malicious flows were assigned at random to ensure that no selection bias was present. As will be reviewed later, this balancing of the data led to far greater accuracy scores, f1 scores, and ROC AUC values.

V. Modeling

A. *Technique Selection*

Initially, this project intended to make use of a variety of different models in addition to random forest and Gaussian Naïve Bayes such as support vector machines, though the results presented by the initial random forest classifier model made exploration into further techniques unnecessary. However, if this project were to be extended further, more models would be utilized to observe the full range of results and to create optimal final results. The initial model types were chosen based on the suggestions of both CTU University and a variety of other sources [1] that have previously fit models to this data.

B. *Data Splitting*

The data was sufficiently large and varied enough that an 80%/20% train test split was seen as sufficient, even after the data set was balanced. The final data set was made up of approximately 900,000 records and split tuning had a very small impact in initial tests. If this project were to be repeated, further usage of k folds validation may be utilized, but the results were strong and had no sign of overfitting, so 80/20 was accepted.

C. *Model Generation*

The initial intention of this project was to generate 4 or 5 different models (RFC, Naïve Bayes, SVM, etc…) and to compare them and select the optimal final result. However, minor tuning within the random forest classifier led to highly accurate models that made further exploration unnecessary aside from as a potential portfolio exercise. An initial Naïve Bayes model was generated, but the results were much worse than the baseline random forest model, so hyperparameter tuning was dubbed unnecessary when a strong alternative model was already in hand.

1) *Per-Scenario Classifiers*

Initially, one model was generated per scenario for a total of 13 random forest classifier models. These models were generated as a baseline and were not tuned in any way so that a greater understanding of the data and potential pitfalls could be uncovered. However, the target feature was still unbalanced and the model metrics had not yet been decided on. As a result, these models were largely useless as a final product. The upside to these models was that the need to balance the target feature was discovered, and the area under the ROC curve was added to the final model's assessment.

2) *Random Forest Classifier*

The random forest classifier model that was generated was found to be the optimal model, as will be covered further in 6a – Result Evaluation. Two functions were generated to test the area under of the ROC curve and the accuracy score of a baseline model varying the max leaf nodes to values of 5, 50, 500, and 5,000. 500 max leaf nodes was the parameter selected as it generated the highest metric values without sacrificing significant runtime. Marginal improvements were present with

5000 maximum leaf nodes, though such an increase significantly increased training time and led to a higher chance of overfitting the model.

RandomizedSearchCV and GridSearchCV were considered for further hyperparameter optimization, but they were deemed unnecessary as the initial metrics were seen as sufficient for the purposes of this project due to the large size of this dataset. If runtime were not an issue, grid searching could be utilized to optimize hyperparameters as such as minimum impurity split, minimum samples per leaf, verboseness, max depth, max features, and a number of other parameters.

*3) **Naïve Bayes***

An initial Naïve Bayes model was generated to see if an alternative model could improve on the random forest classifier model. However, the baseline Gaussian Naïve Bayes model performed as much lower level than the random forest, and as such further tuning was forgone. The reasoning behind this decision is delved into further under the Gaussian Naïve Bayes section of Model Assessment.

D. Model Assessment

The models were assessed utilizing confusion matrices, accuracy scores, ROC AUC scores, and f1 scores. These metrics were chosen as a high ROC AUC places heavy weight on true positive rate, a higher accuracy score places heavy weight on true predictions in general, and f1 is the harmonic mean of the model's accuracy and recall, meaning that a model with high values in each of these metrics performs well regardless of philosophy regarding what is most important (True positive rate, false positive rate, true negative rate, false negative rate).

If the model was not so highly accurate in the chosen random forest model, the most important metric would be the true positive rate. In this case, false positives could be more easily forgiven because IP addresses classified as malware could be later "forgiven", while false negatives would be punished more heavily due to the lack of detection of bots in the network.

*1) **Random Forest Classifier***

The random forest classifier model generated a ROC AUC of 0.950, an f1 score of 0.949, and an accuracy of 0.950, meaning that this model generates highly accurate predictions and the model correctly classifies malicious IP addresses within the test data with an acceptable error rate. Of the 177,855 test records, the model correctly predicted 84,173 true negatives and 83,811 true positives. Alternatively, it generated 3,574 (2% of the test set) false positive predictions, and 5,297 (3% of the test set) false negative predictions, meaning that the model made an accurate prediction 95% of the time.

To understand which features most strongly impacted the model's prediction, Shapley Additive Explanations (SHAP), feature importance, and permutation importance were utilized to create a full picture of the model's inner workings (exploring the "black box").

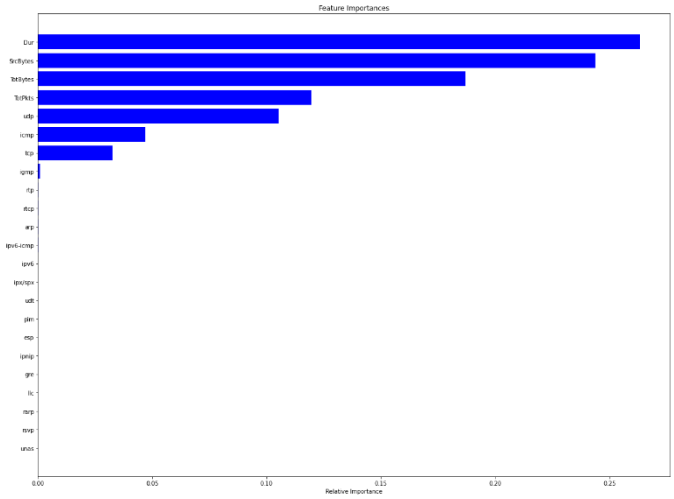

Fig. 1. *Feature Importance Graph for RFC Model*

This graphic calculated feature importance utilizing the random forest's built in feature_importances_ function, which simply illustrates which features most impact the model's predictions. While it is a less sophisticated measure of the most important features, it serves as a solid estimation. This initial figure shows that Duration, Source Bytes, Total Bytes, Total Packets, UDP, ICMP, and TCP are the most important features in predicting whether or not a flow is from a malicious source. By this measure, the majority of the other protocols had minimal impact on the model's predictions. While this lines up relatively well with other blackbox descriptors, more accurate models are yet to come.

Weight	Feature
0.3590 ± 0.0009	SrcBytes
0.2457 ± 0.0012	Dur
0.1693 ± 0.0004	TotBytes
0.1507 ± 0.0005	udp
0.1298 ± 0.0008	icmp
0.0897 ± 0.0007	TotPkts
0.0790 ± 0.0003	tcp
0.0000 ± 0.0000	igmp
0.0000 ± 0.0000	rtp
0.0000 ± 0.0000	rtcp
0.0000 ± 0.0000	arp
0 ± 0.0000	unas
0 ± 0.0000	ipnip
0 ± 0.0000	udt
0 ± 0.0000	gre
0 ± 0.0000	esp
0 ± 0.0000	ipv6-icmp
0 ± 0.0000	rarp
0 ± 0.0000	ipx/spx
0 ± 0.0000	ipv6
… 3 more …	

Fig. 2. *Permutation Importance Graph for RFC Model*

The above figure represents permutation importance values for the primary classifier model. Permutation

importance works by shuffling the values within a single feature, holding all other features constant for each iteration. Theoretically, shuffling a feature should impact the model's predictive accuracy, and the change in accuracy measured there results in permutation importance values. Generally, more important predictive being shuffled will cause the largest impact, meaning that Source Bytes, Duration, Total Bytes, UDP, ICMP, Total Packets, and TCP were the features that were most impactful by this metric. Furthermore, it makes sense that alternative protocols outside of UDP, ICMP, and TCP are given less weight in this model as they make up a smaller portion of the flows observed.

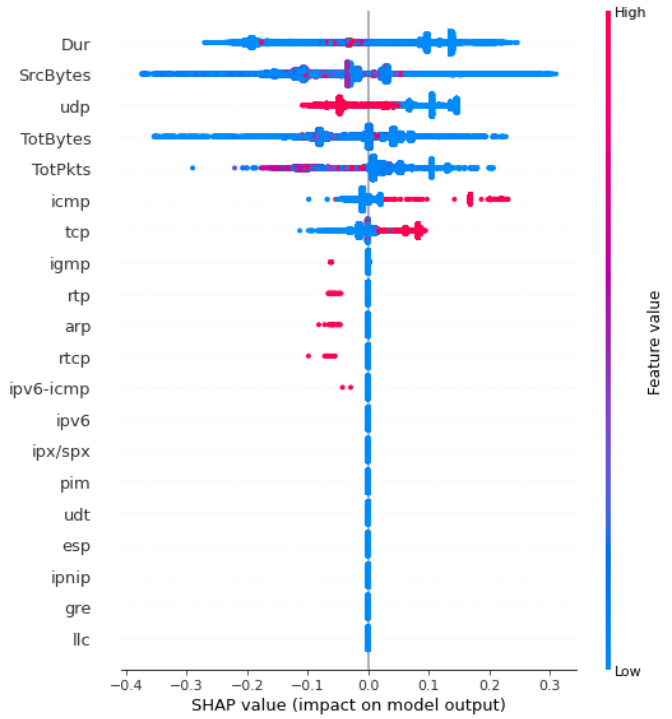

Fig. 3. *SHAP Summary Plot for RFC Model*

SHAP summary plots demonstrate which points have positive or negative impacts on the model's output and to what degree. Horizontal position demonstrates the degree that the specific point affects the output, and the color represents whether the value was high or low for that specific row of data. However, since the majority of the features in this data set were binary, the color is skewed. Furthermore, SHAP summary plots are largely inefficient when dealing with large data sets. Because of this, this SHAP plot is generated on a subset of the test data (25%) and may not be truly indicative of the full data's trend. Regardless, it should serve as a valid estimation.

This plot is relatively consistent with the rest of the model exploration features, indicating that Duration, Source Bytes, UDP, Total Bytes, Total Packets, ICMP, and TCP have the largest impacts on the model's output, with Duration and Source Bytes being the two most important features across the board. Given the coloration issues, it is harder to tell in which direction these individual features impact the model. For example, Duration is nearly completely blue, meaning that most of those values were see as low for their individual rows, yet they impacted the model in both directions. As such, less 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.

2) Gaussian Naïve Bayes

The Gaussian Naïve Bayes model generated a ROC AUC of 0.546, an f1 score of 0.683, and an accuracy score of 0.547. While these metrics could have been improved with further hyperparameter tuning, this was seen as unnecessary for the purposes of this project due to the high metrics observed in the random forest classifier model.

Compared to the Random Forest Classifier model, this model correctly predicted 86,713 malicious flows and 10,506 non-malicious flows. It also generated 78,241 false negatives and 2,395 false positives, making the model effectively useless. Of course, further hyperparameter tuning could have improved this result, but given the overwhelming success of the random forest model, this was unnecessary. The Gaussian Naïve Bayes model was not chosen as the final model, and its poor metrics made exploring the black box unnecessary.

VI. Evaluation

A. Result Evaluation

Overall, this project was a success as a model with 95% accuracy was generated. Minor improvements could be yielded with less concern for runtime and more detailed hyperparameter tuning, but this model is acceptable for the moment. If further improvements were desired for the sake of deployment on a larger scale, this evaluation would need to be recreated with a target accuracy closer to 0.97.

B. Process Review

As a whole, this process would have been greatly expedited if I had the insights that I have now when the project began. Initial models were scrapped because they utilized unidirectional flows, time was wasted on per scenario models that had no chance of ever being utilized, and the final model was generated using sub-optimal functions as opposed to grid searching.

Had the final model not been a basic transformation of the baseline random forest classifier with a balanced training dataset, this project may have taken much more time and effort before an acceptable solution was reached. In the future, more in-depth preparation is necessary to avoid such pitfalls in case an acceptable model is not as apparent. That being said, this project exists as a learning exercise, and in that regard this project was a massive success beyond the basic success of the final model.

C. *Deployment Preparation*

While the final model will not be deployed into a regular workflow, steps could be taken to prepare it for further usage. Utilizing Pyspark or a similar library, pipelines could be generated to automate the data cleaning process including processes such as imputing and feature extraction. This project's cleaning and model creation was done by hand, but for large scale deployment pipelines would be necessary to ensure a consistent process that does not consume too much time and resources. The model training itself could be made into a pipeline or into a function for true automation and replicability with other datasets aside from the CTU-13 data.

VII. Deployment

A. *Project Review*

The goal of this project was simply to generate a classification model capable of classifying malicious net flows with a high degree of accuracy with some forgiveness lent to false positives. In this regard the project was successful, and next steps to expand on this project have been laid out throughout this document.

B. *Theoretical Next Steps*

As previously noted under section 6c, this model would need to have its process normalized via pipelining for largescale deployment. As it stands, the data must be cleaned and the model must be trained by hand, which does not lend itself to time efficient replication in the future. It was previously noted that addition hyperparameter tuning would be necessary for usage beyond a toy example. However, within a pipeline we might be tempted to leave the model as it is to manage runtime. If this project were to continue, a cost benefit analysis of further tuning would be conducted, and then a flask app would be generated. This flask app would then be containerized on Docker and then hosted by amazon web services. However, the deployment process is beyond the scope of this project and would require further research prior to its undertaking.

Acknowledgments

I would like to thank Dr. Eric Goodman and Dr. Dave Eargle for their consultation as this project was completed. Their wisdom and guidance allowed me to push create a better end-product. I would like to thank the members of my M.S. cohort for their support as we have improved our analytical skills together over the past year.

References

[1] García, S., Grill, M., Stiborek, J., & Zunino, A. (2014, June 05). An empirical comparison of botnet detection methods. Retrieved March 27, 2021,from https://www.sciencedirect.com/science/article/pii/S0167404814000923

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

[3] Haghighat, M. H., & Li, J. (2018, November). Edmund: Entropy based attack Detection and Mitigation engine Using Netflow Data. Retrieved February 27th, 2021 from https://www.researchgate.net/publication/329457489_Edmund_Entropy_based_attack_Detection_and_Mitigation_engine_Using_Netflow_Data.

[4] Le, D., & Zincir-Heywood, N., & Heywood, M., Data analytics on network traffic flows for botnet behaviour detection. Retrieved February 27, 2021 from https://ieeexplore.ieee.org/document/7850078

[5] Vishwakarma, A. (2020, May). Network Traffic Based Botnet Detection Using Machine learning. Retrieved March 1, 2021 from https://scholarworks.sjsu.edu/cgi/viewcontent.cgi?article=1917&context=etd_projects