Botnet Detection in Network Traffic Data

Name:	**Parth Gupta**
Registration No./Roll No.:	20201
Institute/University Name:	IISER Bhopal
Program/Stream:	EECS
Problem Release date:	August 17 2023
Date of Submission:	November 19 2023

1 Introduction

Botnets refer to a collection of computers that have been compromised by malware or are under the remote control of an adversary. These compromised systems, when commanded by adversaries, can launch coordinated attacks on a server. This coordinated effort often leads to Denial of Service (DoS) attacks, overwhelming servers with an excessive number of requests and causing them to fail in handling legitimate clients.

Our project focuses on classifying different botnet types using Single-Flow-Time-Series (SFTS) data derived from internet packets. The dataset encompasses 84 features, comprising default features and distribution-based, frequency-based, and time-based features derived from the default ones. Our objective is to categorize these features into six distinct classes: "donbot," "fastflux," "neris," "qvod," "rbot," and "clear."

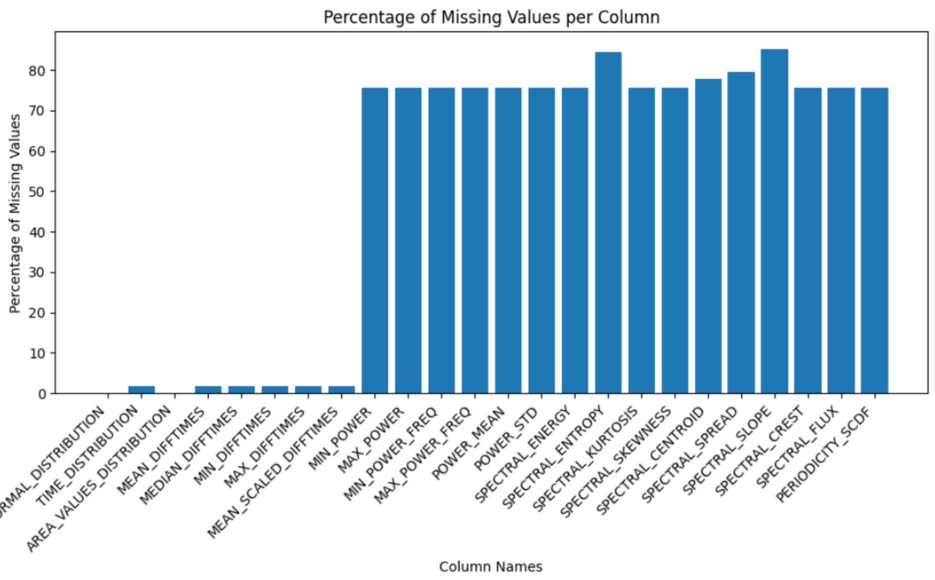

Figure 1: Overview of features with percentage of missing data.

2 Methodology

Initial exploration of the dataset revealed a tonne of missing data. Out of the 83 features, many have more than 75% of data missing. As depicted in Figure 1, we can clearly see the amount of missing data in some features. (Note that only features with at least one missing value are shown in the graph).

There was a need for imputation of these missing values, so we implemented various strategies. In a later section, we reported the evaluation metrics we got from these different strategies. Figure 2 clearly explains the complete process flow followed during the project.

Firstly, we will be imputing the values using mean, median and mode. Then we will remove the features having more than 75% data missing[1] and reapply those imputation methods. Then we will compare the various evaluation metrics between them.

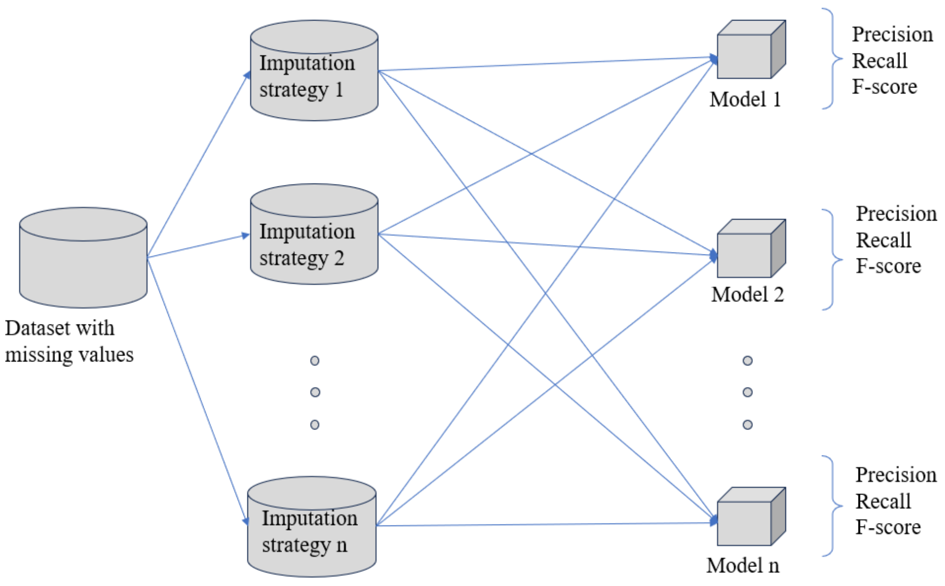

Figure 2: Explanation of the methodology.

Moreover, for feature selection, we will use the Mutual-Info feature[2] selection technique. Then, we will compare our results with and without the feature selection.

3 Experimental Setup

The code on GitHub can be found here. As discussed above in Figure 2, we will be reporting the macro averaged precision, recall and f-score. We have created separate functions that are capable of running multiple models simultaneously and return the 'best_params' that should be used for training those specific models. It also outputs the evaluation metrics of those models for each of the imputed datasets.

Let's go over through them to help better understand the results. ('IS' is an alias for Imputation Strategy)

IS number	Explanation
1	Replace missing values with mean
2	Replace missing values with median
3	Replace missing values with mode
4	Remove features with more than 70% missing values then impute with mean
5	Remove features with more than 70% missing values then impute with median
6	Remove features with more than 70% missing values then impute with mode

What's more, is that we will be heavily using Python's Multiprocessing capability to train multiple models concurrently on different cores of the CPU. This proved essential in decreasing the time it takes to train multiple models on multiple different datasets for multiple iterations using Grid Search. Overall, the total execution time with multiprocessing took more than 2 days on our local machine.

4 Results and Discussion

The tables shown below report the macro averaged Precision, Recall and F-Score (in that sequence) of each model on each dataset (imputed using a different strategy). The top 6 tables are metrics without using feature selection techniques.

IS 1			
LR	KNN	RF	DT
0.15623	0.968588	0.999571	0.981638
0.166661	0.822402	0.963252	0.999636
0.161277	0.878506	0.979544	0.990261

IS 2			
LR	KNN	RF	DT
0.156225	0.974425	0.966092	0.984702
0.166667	0.81169	0.943318	0.999848
0.161277	0.875314	0.954201	0.991914

IS 3			
LR	KNN	RF	DT
0.156225	0.980735	0.999425	0.981638
0.166667	0.819806	0.946586	0.999636
0.161277	0.882432	0.970699	0.990261

IS 4			
LR	KNN	RF	DT
0.15623	0.984177	0.966238	0.984691
0.166667	0.925203	0.959985	0.996368
0.16128	0.951981	0.963046	0.990153

IS 5			
LR	KNN	RF	DT
0.15623	0.984177	0.966027	0.981638
0.166667	0.925203	0.943318	0.999636
0.16128	0.951981	0.954168	0.990261

IS 6			
LR	KNN	RF	DT
0.15623	0.984177	0.999636	0.981638
0.166667	0.925203	0.949854	0.999636
0.16128	0.951981	0.972455	0.990261

The six tables below are evaluation metrics after using feature selection using mutual_info_classifier. Note that all the values in tables are obtained after using the best parameters for each classifier using Grid Search.

IS 1			
LR	KNN	RF	DT
0.870442	0.968639	0.999848	0.984821
0.435281	0.889445	0.999642	0.989626
0.468143	0.925636	0.999745	0.986714

IS 2			
LR	KNN	RF	DT
0.866781	0.967869	0.999777	0.984843
0.420717	0.889423	0.966308	0.996586
0.455992	0.925241	0.981191	0.990337

IS 3			
LR	KNN	RF	DT
0.875786	0.967869	0.999777	0.984843
0.420316	0.889423	0.966308	0.996586
0.455992	0.925241	0.981191	0.990337

IS 4			
LR	KNN	RF	DT
0.866781	0.967853	0.996784	0.984837
0.420717	0.889276	0.966308	0.993318
0.455992	0.925158	0.979679	0.988651

IS 5			
LR	KNN	RF	DT
0.866781	0.967853	0.996509	0.984832
0.420717	0.889276	0.96304	0.99005
0.455992	0.925158	0.977923	0.986931

IS 6			
LR	KNN	RF	DT
0.866781	0.967853	0.996509	0.984837
0.420717	0.889276	0.96304	0.993318
0.455992	0.925158	0.977923	0.988651

For the feature selection, we have taken the top 30% of the most important features to train and evaluate the models.

A few artefacts are noteworthy:

- Logistic regression had very poor performance in general.
- Feature selection increased the F-score or Logistic regression by 40% (and precision by about 70%).
- Overall feature selection with 30% features did not cause significant loss in performance of classifiers.

- For this case, Random Forest is working the best.
- Imputation strategy 1 with Random forest and feature selection yields the highest f-score among all models.

5 Note

Kindly note that the testing dataset also contains multiple missing values which were imputed according to the best strategy discussed in the above section. After the testing dataset was complete, only then output labels were printed.

References

[1] R. sivakani and g. ahmad ansari, "imputation using machine learning techniques," 2020 4th international conference on computer, communication and signal processing (icccsp), chennai, india, 2020, pp. 1-6, doi: 10.1109/icccsp49186.2020.9315205.

[2] Wei-chao and chih-fong tsai. missing value imputation: a review and analysis of the literature. artifcial intelligence review (2020) 53:1487–1509 https://doi.org/10.1007/s10462-019-09709-4, 2019.