TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies
Department of Software Science

[IAY70LT]
Liubomyr Kushnir IVCM184596

BENCHMARKING OF POST-HOC LOCAL INTERPRETABILITY METHODS FOR CLASSIFYING MALICIOUS TRAFFIC

Master's Thesis

Supervisor: Hayretdin Bahsi
Ph.D.
Co-Supervisor: Sven Nõmm
Ph.D.

Tallinn 2020

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

[IAY70LT]
Liubomyr Kushnir IVCM184596

POST-HOC TÕLGENDAMISMEETODITE VÕRDLUSUURING PAHATAHTLIKU VÕRGULIIKLUSE KLASSIFITSEERIMISEKS

Magistritöö

Juhendaja: Hayretdin Bahsi
Ph.D.
Kaasjuhendaja: Sven Nõmm
Ph.D.

Tallinn 2020

Author's declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, the literature and the work of others have been referenced. This thesis has not been presented for examination anywhere else.

Author: Liubomyr Kushnir

2020-08-04

Abstract

Machine learning models are becoming more popular these days, and with this popularity comes the requirement to interpret the nature of outputs of these models. In this thesis presented a comparison of LIME(Local Interpretable Model-Agnostic Explanations) and SHAP(SHapley Additive exPlanations) post-hoc local interpretation algorithms. They are evaluated based on how well they perform an interpretation of machine learning models that classify malicious IoT botnet traffic from various devices to create a benchmark. The research consists of computing several metrics on selected data and analyzing and comparing the results. A comparison of LIME and SHAP has been made before in forums and articles, but there is no specific dashboard with metrics and interpretation of results. Applying this interpretation algorithms in the field of cybersecurity, on IoT botnet dataset, is also new research which will help identify and classify attacks.

This thesis is written in English and is 78 pages long, including 5 chapters, 42 figures, and 17 tables

Annotatsioon

Masinõppe mudelid muutuvad iga päevaga populaarsemaks, sellega kaasneb vajadus interpreteerida mudelite väljundi olemust. Käesolevas magistritöös võrreldakse kaht post-hoc lokaalset interpreteerimisalgoritmi - LIME (Local Interpretable Model-Agnostic Explanations) ja SHAP (SHapley Additive exPlanations). Nende võrdlemiseks kasutatakse masinõppe mudeleid, mis klassifitseerivad pahatahtlikku IoT robotvõrgu liiklust. Uurimistöös väljapakutud lahendus koosneb mitme mõõdiku arvutamisest ja tulemuste analüüsist ja võrldusest. LIME ja SHAP algoritme on võrreldud varem foorumites ja artiklites, kuid puudub konkreetne mõõdikute ja tulemuste näidikupaneel. Nende võrdlusalgoritmide rakendamine IoT robotvõrgu liikluse andmetel on uus uurimistöö, mis aitab tuvastada ja klassifitseerida rünnakuid.

Lõputöö on kirjutatud English keeles ning sisaldab teksti 78 leheküljel, 5 peatükki, 42 joonist, 17 tabelit.

List of abbreviations and terms

ASCII American Standard Code for Information Interchange

KNN K-Nearest neighbours

LIME Local Interpretable Model-Agnostic Explanations

SHAP SHapley Additive exPlanations

SVC Support Vector Classifier

Table of Contents

1 Introduction **13**

1.1 Contribution and Novelty 13

1.2 Problem statement 15

1.2.1 Classifier selection 16

1.2.2 Feature selection 16

1.2.3 LIME/SHAP explanation analysis 16

1.2.4 Comparing LIME/SHAP explanations with Decision Tree 16

1.2.5 Metrics 17

1.3 Workflow 17

2 Background **18**

2.1 Literature review 18

2.2 Local Interpretable Model-Agnostic Explanations 20

2.3 SHapley Additive exPlanations 21

2.4 Decision Tree. Explanation path interpretation method 22

2.5 Unified way to interpret explanation results 22

3 Methodology **24**

3.1 Dataset 24

3.2 Classifier selection 25

3.3 Dataset feature selection 25

3.4 LIME/SHAP explanation analysis 26

3.5 LIME and SHAP and Decision Tree comparison 27

3.6 Metrics . . . 27
3.6.1 Consistency . . . 28
3.6.2 Novelty . . . 28
3.6.3 Sensitivity . . . 28
3.6.4 Stability . . . 28
3.7 Levenshtein distance . . . 28

4 Solution **30**

4.1 Classifier selection . . . 30
4.2 Feature selection . . . 32
4.3 LIME/SHAP explanation analysis . . . 33
4.3.1 LIME Feature selection . . . 33
4.3.2 SHAP Feature selection . . . 37
4.3.3 LIME Feature weights . . . 41
4.3.4 SHAP Feature weights . . . 46
4.3.5 Comparing LIME and SHAP feature weights . . . 49
4.3.6 Explaining points close to decision boundary . . . 57
4.4 LIME and SHAP and Decision Tree comparison . . . 62
4.5 Metrics . . . 65
4.5.1 Consistency . . . 65
4.5.2 Novelty . . . 67
4.5.3 Sensitivity . . . 68
4.5.4 Stability . . . 69
4.6 Levenshtein distance . . . 71

5 Conclusion **73**

References **76**

List of Figures

1 A sample output of LIME interpretation 23

2 A sample Decision Tree model 32

3 ack feature selection 36

4 benign feature selection 36

5 combo feature selection 37

6 udp feature selection 37

7 ack feature selection 39

8 junk feature selection 39

9 scan feature selection 40

10 udp feature selection 40

11 `MI_dir_L0.01_weight` feature weights 43

12 `H_L0.01_weight` weights sample within tcp class 45

13 `HH_L0.1_magnitude` weights sample for "ack" class 48

14 `MI_dir_L0.1_weight` weights sample for "scan" class 49

15 `HH_L3_weight` for "junk" class 56

16 `H_L0.01_mean` for "udp" class 57

17 "udp" data points location 58

18 "combo" samples prediction probability 58

19 Combo samples location 59

20 Close to decision boundary 60

21 Far from decision boundary 60

22 Close to decision boundary 60

23 Far from decision boundary 60

24 Close to decision boundary 61
25 Far from decision boundary 61
26 Close to decision boundary 61
27 Far from decision boundary 61
28 `MI_dir_L0.1_weight` feature order-selection 63
29 `HH_L3_weight` feature order-selection 64
30 `H_L0.01_mean` feature order-selection 65
31 Decision Tree model 66
32 KNN model 66
33 Decision Tree model 67
34 KNN model 67
35 LIME explanation 68
36 SHAP explanation 68
37 LIME explanation 69
38 SHAP explanation 69
39 LIME benign explanation 69
40 LIME benign explanation 69
41 SHAP benign explanation 70
42 SHAP benign explanation 70

List of Tables

1 LIME vs SHAP [7] 14

2 Classifiers stats for different distribution of benign-malicious data 30

3 Top 10 Fisher score features 32

4 LIME feature selection comparison 34

5 LIME features for every class 35

6 Top 5 features picked by SHAP 37

7 SHAP features for explaining every class 39

8 Top 5 LIME features weights 42

9 LIME feature weights for every class 45

10 SHAP feature weights for every class 47

11 Top 5 LIME "all class" explanations features compare 50

12 Top 5 SHAP "all class" explanations features compare 51

13 Comparing LIME features for "one class" explanations 53

14 Comparing SHAP features for "one class" explanations 55

15 Stability metric features data 70

16 Levenshtein distance within a class 72

17 Comparison table 73

1 Introduction

Classification algorithms gained popularity in the last two decades[1] because the computer's computation capabilities have grown, and the cost has significantly fallen. The variety of classifiers is relatively wide. However, for some of the problems, a good classifier is not enough, and an interpretation of classification is required. One such problem is detecting and classifying malicious traffic. In cybersecurity, analyst requires explanation because the influence of features means more than model output. For example, if model prediction probability is not high enough - without explanation, the analyst might not be able to distinguish between malicious traffic and benign. Classifying botnet is only half of the problem, the classification problem has been done, solved and described in several papers like [2] and [3]. At the start of this year, there has been another publication of detecting botnet using reinforced learning in [4]. Classification of botnets has already reached a satisfactory accuracy level - above 97%. Now it is the time to turn to another half or the problem - interpretability of these classifiers.

1.1 Contribution and Novelty

This thesis targets benchmarking of interpretability algorithms. "Benchmark is the act of running a computer program, in order to assess the relative performance of an object, normally by running a number of standard tests and trials against it"[1]. After reviewing several papers and articles, there was found a lack of a good comparison of LIME and SHAP in terms of separate metrics. In the article [5] author explains details of each of these interpretation methods, talks about advantages and disadvantages, but does not have a clear side by side comparison. In paper [6] presented a nice overview on sensitivity metric for LIME and SHAP. This is one of the metrics used in this thesis, so results are compared. However, one metric is not enough to compare these two algorithms. The article [7] contains study about LIME, ELI5, SHAP and InterpretML. It lists advantages and disadvantages of LIME, as well as presents a table with a comparison of LIME and SHAP. Table 1 is one of the best comparison found in the literature.

[1] https://en.wikipedia.org/wiki/Benchmark_(computing)

	LIME	SHAP
1	LIME is model-agnostic, and it can be applied to any machine learning model.	SHAP guarantee properties like consistency and local accuracy.
2	LIME assumes that the local model is linear or Decision tree.	Shapley values consider all possible predictions for an instance using all possible combinations of inputs.
3	LIME Output is very fast even for large dataset.	SHAP value calculation is very time expensive as it checks all the possible combinations.
4	LIME value is difference between the prediction with and without a variable	SHAP value is contribution of a variable to the difference between the actual prediction and the mean prediction.

Table 1. LIME vs SHAP [7].

The goal of this thesis is to create a comparison of LIME and SHAP using pre-defined metrics. Currently, there is no benchmark for these interpretability algorithms, that will compare metrics or explanations. This work is targeting the cybersecurity area and will allow cybersecurity analysts to understand the specific of LIME and SHAP better. The application of these interpretability methods in the cybersecurity area has been done before and described in the paper [8], but without a clear comparison. Having such a benchmark, one can more easily make a decision on which method to choose for their problem, and which method is better applicable to a specific case. This benchmark is constructed for cybersecurity area and can be applicable only in it. However, some of the metrics, that were computed to construct the benchmark does not depend on the specific dataset, thus can be used and applied in other domains.

All experiments and calculations are performed on IoT botnet dataset [9]. This dataset addresses the lack of public botnet datasets. It contains traffic data from the IoT(Internet of Things) devices. The data is gathered from 9 commercial devices authentically infected

by Mirai and BASHLITE [9]. These devices are two different doorbells, four different security cameras, a baby monitor, and a thermostat. In the thesis, there is not much focus on training classifiers, as this task was relatively easy to solve and as soon as a model with decent prediction accuracy was trained - it was time to move forward to interpretation. Post-hoc local interpretability methods solve the problem of interpretation. These explainers provide data about sample features that influence the output of sample prediction.

1.2 Problem statement

Constructing benchmark required gathering statistics about different metrics and comparing metrics with each other. These metrics are consistency, novelty, sensitivity, stability, feature selection and weights, and Levenshtein distance. Each from the list will be described in detail later. All the data for metrics gathered from various experiments, which involved explanations of different sets of samples. A Decision Tree explanation path has been used as a third alternative for some experiments. Its results are not separated as another metric, because it has nothing to do with comparing LIME and SHAP. But having another intuitive explanation of a sample provides another perspective on certain problems. It is not an interpretation algorithm, however let us not forget the main idea of interpretation - to explain the outputs of a "black box" model. "Interpretability is defined as the ability to explain or to provide the meaning in understandable terms to a human" [10], and explanation path satisfies this definition. Hence, it is used in experiments where applicable.

Explanations of LIME and SHAP contain similar information, but it is presented in different ways and structures. One of the issues was to interpret the results of explainers in a unified way so that they could be compared with each other. The explanation path was used where applicable, and its results had to be represented in the same form as LIME and SHAP. A detailed description of the differences of inputs and outputs of each algorithm that was used for this research will be described later.

This study is solving the problem, which consists of several sub-problems:

1.2.1 Classifier selection

The machine learning classifier is an essential part of this thesis, as it is responsible for correct predictions of the interpretation algorithm. Interpretation is a computationally intense task. To save time and computational resources the model is chosen to have low complexity. Model accuracy is above 90% which is acceptable level for current experiments, however very far from ideal.

1.2.2 Feature selection

Feature selection is not the part of main research and takes no part in comparing LIME with SHAP. It is a separate work, which helps to understand the most important features from the initial 115 features of the dataset. Also, a fewer number of features decrease the computational time of most calculations, as it decreases dimensions of the dataset.

1.2.3 LIME/SHAP explanation analysis

The goal is to gather and analyze LIME/SHAP interpretation results of various sets of samples. Which features are mostly picked by LIME/SHAP to explain instances? What are those feature weights? Is there variance of feature order and feature weights between different classes of the same dataset chunk? If there is - what is it? Analyzing points that are closer and further away from the decision boundary: is there any difference is classification? Is there any difference is the explanation? If there is - how different are the features, feature order, and feature weights?

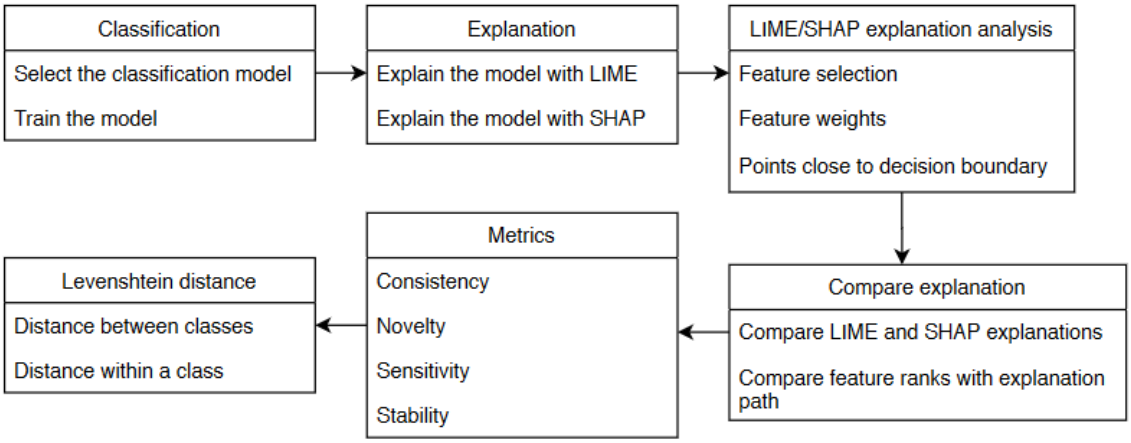
1.2.4 Comparing LIME/SHAP explanations with Decision Tree

Analyzing the explanation of LIME, SHAP, and Decision Tree explanation path is important because it allows to see how much a simple explanation of Decision Tree differs from interpretability methods. Can it be useful, what can it provide and how bad it is comparing to LIME and SHAP. As was mentioned earlier, minimal statistics can be collected from the Decision Tree classifier, but where applicable is compared with LIME and SHAP.

1.2.5 Metrics

There are several metrics, defined in the paper [11]. Some of them are selected for benchmarking and statistics is gathered about each of them. Selected metrics are - consistency, novelty, sensitivity, and stability. Details about each of these metrics are described in the Metrics section. Levenshtein distance is used as another metric of comparing the difference between two explanations.

1.3 Workflow



The workflow consists of several steps: select and train classification model; explain this model with both LIME and SHAP; analyze explanations and compare them; collect data about specific metrics and calculate Levenshtein distances between explanations. This workflow will lead to the comparison of various results, gathered for LIME and SHAP.

2 Background

2.1 Literature review

The topic of this thesis is comparing two interpretability methods. Metrics and measures are required, and paper [12] provides a good overview of two major difficulties in the measure of interpretability. Firstly, distinct terms are used in the literature. They have to be separated into the ones used as strict synonyms (e.g. understandability and comprehensibility) and the ones that depend on interpretability for specific cases. These are related to distinct problems (e.g. justifiability and usability). Secondly, papers in the literature can be divided into comparisons of the interpretability of models and representations. Where representations are comparisons based on mathematical heuristics or user-based surveys.[13]

The definition and requirement of interpretability are well explained in the paper [10]. The study is about methods for explaining various types of models. It also touches the need for interpretation, describing several cases where "black box" models had unreasonable results or caused business impact. A computer program for screening job applicants that was found to unfairly discriminate against ethnic minorities and women. By inferring information from surnames and place of birth, it was lowering their chances of being selected for interview. [14] The journalists of propublica.org have shown that the COMPAS score, a predictive model for the "risk of crime recidivism" (proprietary secret of Northpointe), has a strong ethnic bias [1]. Another example is related to Amazon.com. In 2016, the software used to determine the areas of the US to which Amazon would offer free same-day delivery, unintentionally restricted minority neighborhoods from participating in the program (often when every surrounding neighborhood was allowed) [2].

With respect to credit bureaus, it is shown in [15] that banks providing credit scoring for millions of individuals, are often discordant: in a study of 500 000 records, 29% of consumers received credit scores that differed by at least fifty points among three major US banks (Experian, TransUnion, and Equifax). Such a difference might mean tens of thousands of dollars over the life of a mortgage.[16] Interpretation of model results might help to prevent such problems, or debug the model and help to correct them.

[1] https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing
[2] https://www.businessinsider.com/how-algorithms-can-be-racist-2016-4

Study also includes terms of interpretability dimensions, like: global and local interpretability, time limitation and nature of user expertise [17]. The first one - global and local interpretability is touched in this thesis, as the comparison is between the method that provides global interpretability and the one that provides local. When a model is completely interpretable, i.e., we are able to understand the whole logic of a model and follow the entire reasoning leading to all the different possible outcomes. This is the case of global interpretability. On the other hand, local interpretability is the situation in which it is possible to understand only the reasons for a specific decision: only the single prediction/decision is interpretable. [17] LIME is considered to be a local interpreter, because it explains a single instance, and executes all calculations for one sample at a time. SHAP requires a set of sample, to provide a full explanation of the model, as well as samples themselves. If a separate explanations are required - they need to be extracted from the explanation set, produced by SHAP. That is why SHAP is a global interpreter.

The dataset for this thesis contains botnet traffic, and to have a starting point - a research about similar data is studied. A paper [3] is about the study of classifying botnet attacks on internet relay chat server. Four out of six classification models, described in the paper was used in this study. Classification accuracy of these models in the paper is approximately 99% which is considered high, and using the same models for this study had similar results. These results are presented later in the Table 2. However, this study addresses only the classification, not interpretation. Nevertheless this thesis involves creating classification models and the pool of models used in the paper [3] provides a good overview.

A paper [18] describes an example of the interpretation of malware detection model. This is very relative to this topic, however no post-hoc methods are used there. Instead, explanation is being extracted from the neural network during the feed forward procedure, or inference. This is not the method used by LIME and SHAP, as they are post-hoc model-agnostic explainers. Deep learning models are much more complex, compare to the ones used in this thesis. For training a deep learning model for malware detection authors used Xeon E5-2697 CPUs, 384 GB memory, and four Nvidia TitanXP graphics cards [19]. This a a very powerful hardware.

A paper [20] is an introduction to model-agnostic explanation and LIME algorithm. It is written by the same authors, who made a paper [21]. This is an introduction to model-agnostic explanations based on if-then rules, which is called anchors. An anchor expla-

nation is a rule that sufficiently "anchors" the prediction locally – such that changes to the rest of the feature values of the instance do not matter [22]. A paper [23] contains study about SHAP, the second interpretability method used in the thesis. SHAP values attribute to each feature the change in the expected model prediction when conditioning on that feature. They explain how to get from the base value, that would be predicted if we did not know any features to the current output [24]. The computation of SHAP values is a complicated task, thus they are often approximated. SHAP values can be estimated directly using the Shapley sampling values method[25] or equivalently the Quantitative Input Influence method[26]. This study involves several types of SHAP, like Kernel SHAP, Linear SHAP, Low-Order SHAP, Max SHAP and Deep SHAP. These are different version of SHAP for various use-cases. In this thesis a Kernel SHAP was used, which functionality is similar to LIME. SHAP feature weighting kernel uses linear regression to estimate SHAP values, while LIME uses linear regression to locally classify samples around the target instance.

Possible legal issues related to interpretation listed in the paper [27]. According to GDPR, an explainer that operates with person identifiable information must have the right for interpretation. However, this study operates with a network traffic, thus has no concerns of breaching a person's privacy.

A benchmark, or other types of comparisons between LIME and SHAP interpretability results was not found in the literature. These algorithms are listed and applied separately in several studies. Some articles like [5] have theoretical compare, without gathering data about specific metrics or feature explanations.

2.2 Local Interpretable Model-Agnostic Explanations

LIME stands for Local Interpretable Model-Agnostic Explanations. It is an interpretation algorithm which uses model-agnostic approach. Model-agnostic means that LIME extracts post-hoc explanations by treating the original model as a blackbox. This involves learning an interpretable model on the predictions of the black box model, perturbing inputs and seeing how the black box model reacts.[21] LIME executes the following steps to build the interpretation:

1. Permute data - The first step is to take all the data around the point, which we need to interpret and permute feature values to generate some new fake data.

2. Calculate distance between permutations and original observations.
3. Make predictions on new data using the initial model.
4. Pick N features best describing the complex outcome from the permuted data.
5. Fit a simple model to the permuted data with N features and similarity scores as weights.
6. Feature weights from the simple model make explanations for the complex models' local behavior.

Before computing interpretation LIME builds an explainer model. LIME takes several parameters to build the explainer, and those are dataset, feature names set, classification model, and classes names. Explanation of the single instance of data is produced by this explainer. The input to the algorithm is instance features values and a limit of features to explain. The output is an object that contains basic information about Ridge regression[28] parameters and an instance explanation itself. Explanation consists of the feature-weight map, picked for this particular instance. The weight in this context is a value assigned to a feature by explainer, which indicates how much does the feature pushes the output of the classifier to the positive or negative side. Feature order in explanation is based on the absolute value of feature weight.

The explanation is not a single set of features and weights. It is an array of explanations for every possible class. This means that for a single given instance, LIME produces a set of explanations. Sometimes, the actual label of an instance differs from the one predicted by the classifier. In such cases, it is useful to have all explanations computed, so one can compare how much one of the original class differs from the one predicted by a classifier.

2.3 SHapley Additive exPlanations

SHAP stands for SHapley Additive exPlanations[29]. It is also a post-hoc interpretation method that performs interpretation similar to LIME. The main steps are the same - generate new data, fit a simple model and use it for building feature weights [23]. SHAP has a few differences - it uses a genetic algorithm to generate new data, and it uses the Shapley values method [30] from game theory to compute the weights of the features. The computation of feature X weight looks like this:

- Get all subsets of features that do not contain X
- Compute the effect on our prediction of adding X to all those subsets

SHAP has a similar interface, and one needs to provide a dataset and class names to train the explainer, but the outputs are different. Unlike LIME, the dataset should be normalized. After SHAP builds an explainer - it takes a set of instances and calculates the explanation for each of them based on the other instances. This means that SHAP cannot be used to explain a single instance. Feature weights are computed based on the input set, and if the input is a single instance - all features will have zero influence.

2.4 Decision Tree. Explanation path interpretation method

Decision Tree is a classifying model. It is not an explainer, thus does not provide informative explanation. Decision Tree explanation path can be used to explain current classification. The explanation path of Decision Tree is a set of nodes, which form a path from the root node to the one with instance class. The number of nodes and leaves is configured before the classifier is trained. The explanation path shows significant features that are used during classification. It does not have the essential feature weights, and the order of features is basically the structure of the tree.

2.5 Unified way to interpret explanation results

The simple structure of LIME interpretation results is easy to understand and contains all the necessary information for instance explanation. This format of results was chosen to be the unified way of interpret the explanations of both methods. The initial explanation results of LIME and SHAP look different, and it is not easy to compare the results. Thus, one of the method's results must be transformed to the type of another method. The goal of unifying explanation results is to understand how LIME library builds explanation graphs and transform SHAP result into LIME format.

An example in presented in Figure 1. In this thesis, all outputs are programmed to look like this one, so it is essential to understand them. On the top of the graph is the class' name, or sometimes class id from one to eight. The x-axis is the feature weight, and the

y-axis is the feature name. For a better view, positive and negative weights are colored in green and red. Features along the y-axis are sorted by descending based on their absolute weight value. The number of features usually ten. This constant was used through all the experiments in this thesis, and the number of features in a single explanation is up to ten. SHAP does not have a minimal feature count requirement, and occasionally explains instances with less than ten features.

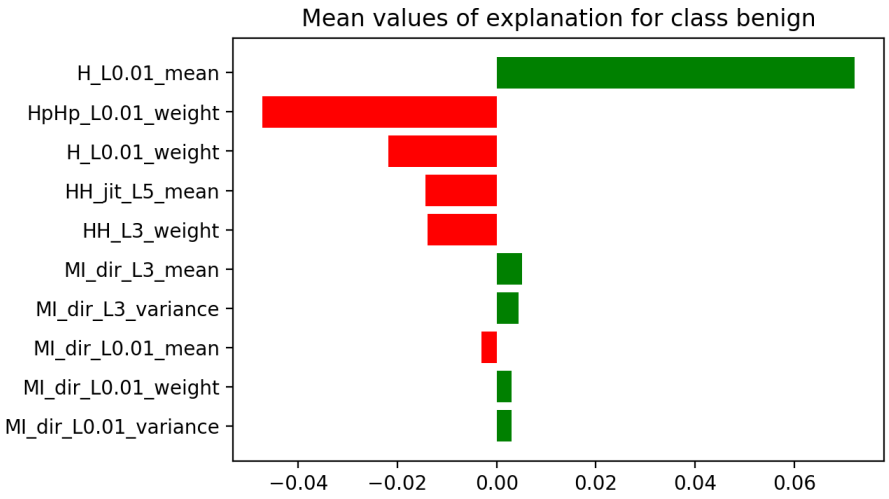


Figure 1. A sample output of LIME interpretation .

3 Methodology

3.1 Dataset

The thesis research process consisted of several experiments conducted on a dataset of 7062606 samples, which have 115 features. The sample is the data of traffic during normal device activity, during Gafgyt and Mirai attacks. For each of these classes, they are divided into subclasses which are located in separate files with specific types of traffic [31]:

- Gafgyt:
  1. Scan: Scanning the network for vulnerable devices
  2. Junk: Sending spam data
  3. UDP: UDP flooding
  4. TCP: TCP flooding
  5. COMBO: Sending spam data and opening a connection to a specified IP address and port
- Mirai:
  1. Scan: Scanning the network for vulnerable devices
  2. Ack: Ack flooding
  3. UDP: UDP flooding
  4. Syn: Syn flooding
  5. UDPplain: UDP flooding with fewer options, optimized for higher PPS
- Benign: normal traffic during idle work of the device

During most experiments test dataset has been divided into eight classes. Some classes have Mirai and Gafgyt subclasses merged. Only the same attack types have been merged into a single class and it was done in order to train a machine learning model to distinguish attack types, not the specific malware type. Each of these classes represent random samples from specific files, and refer to a specific type of traffic.

1. ack: Mirai ack flooding

2. syn: Mirai syn flooding
3. udp: Gafgyt udp flooding, Mirai UDP and UDPplain files
4. tcp: Gafgyt tcp flooding
5. junk: Gafgyt spam data
6. combo: Gafgyt COMBO files
7. scan: Mirai scanning
8. benign: benign traffic of all devices mixed

3.2 Classifier selection

To select the classifier for following experiments - four basic classification algorithms are picked. These are Support Vector Classifier(SVC) [32], K-Nearest Neighbors(KNN) [33], Decision Tree [34] and Logistic Regression [35]. Listed classifiers are selected, because they we used in paper [3] for botnet classification before. Also, they are a popular solution for classification, they are easy to understand, easy to implement and easy to train. Based on the standard metrics like accuracy, precision and recall, the model is selected. For selecting the best classifier the dataset was divided into two classes, which are benign and malicious traffic.

3.3 Dataset feature selection

The paper [36] contains various feature selection techniques, but only one has to be chosen. Chi-squared χ^2 [37] and Fisher score[38] algorithms were chosen, because they are relatively accessible and their implementation is easy to use. Fisher score is selected for feature selection because its results are prevalent among feature selection terms. Since this is a side research and not the part of benchmark - no extra work was dedicated to choose the feature selection algorithm. The paper [8] contains Fisher score results for the same dataset. These results match with the ones described in Feature selection section.

"The key idea of the Fisher score is to find a subset of features, such that in the data space spanned by the selected features, the distances between data points in different classes are as large as possible, while the distances between data points in the same class are as small as possible" [39].

Let μ_k^j and σ_k^j the mean and standard deviation of k-th class, corresponding to the j-th feature. Let c be the number of classes, and n_k is the number of samples, which belong to the k-th class. Let μ^j and σ^j denote the mean and standard deviation of the whole data set corresponding to the j-th feature. Then the Fisher score of the j-th feature is computed below [39],

$$F(x^j) = \frac{\sum_{k=1}^{c} n_k(\mu_k^j - \mu^j)^2}{\sum_{k=1}^{c} n_k(\sigma_k^j)^2}$$

3.4 LIME/SHAP explanation analysis

In this section, various experiments with LIME and SHAP are conducted. Terms "all class" and "one class" results are used to distinguish results.

"All class" is a result of the experiment conducted on the entire dataset. Samples for such experiments are randomly selected without any criteria or filters from the dataset.

"One class" means that samples for the current experiment have been taken from a single class data. Usually, such experiments are being compared with "all class" results during the research of feature weights and feature order. This is done in order to see the difference in results of the same experiment while using samples of a single class versus samples from different classes.

For each explainer, several experiments have been conducted to gather the data about the following:

1. Explainer feature selection. Explainers select a set of features for an interpretation of an instance. These features form the explanation, so which one of them is being selected is an important statistic. Which features are being selected the most for "all class" explanations? Which features are being selected the most for "one class" explanations? Does LIME and SHAP select the same features for explaining particular classes?

2. Feature weights. Weights are another important part of the explanation. This task is about analyzing feature weights values, how are the distributed. Is there a difference in feature weights for different classes. What is the difference in feature weights of LIME and SHAP for the same samples? Do "all class" results differ from "once class" results?

3. Explaining points close to the decision boundary[1]. A subset of samples is selected, closer to the decision boundary. Explanations of these sample are compared with the samples from the main distribution. Statistics is selected from this comparison, that can be summarized and plotted.

3.5 LIME and SHAP and Decision Tree comparison

Decision Tree is a classifier for almost all experiments. LIME and SHAP use it as classification model only, for them it is a "black box" model, like any other. On the other hand, for this study Decision Tree also used to collect explanation path. Since Decision Tree is used as a classifier, all the explanation paths for every experiment is being computed during classification and interpretation by LIME or SHAP. Thus, after a sample had been explained - not only interpreters' results are ready, but an explanation path as well, which is extracted from the Decision Tree classification. The explanation path does not have feature weights, it only contains the used features and their order. Other Decision Tree data, like Gini Index or Entropy cannot be used to compare with LIME or SHAP feature weights. These are measures of disorder and how often a random sample is incorrectly classified, while interpreter feature weights measure how does a particular feature influence overall classification result for the sample. Thus, only feature selection and feature order is compared with LIME and SHAP.

3.6 Metrics

There are a few metrics that are applicable to thesis experiments. These are existing metrics, their definitions have been taken from the [11] and does not contain any specific formulas.

[1] https://courses.cs.ttu.ee/w/images/6/69/Lecture_13_Trace_Explain_Interpret_2020.pdf

3.6.1 Consistency

Consistency metric is about comparing the explanation of the same sample while using different classification models. These models should be trained on the same dataset and have similar accuracy. If the explanations of compared models are similar, then the explainer is considered highly consistent[40].

3.6.2 Novelty

Novelty metric is about comparing the explanations of an instance from the training data distribution, and an instance which is far away from the training data distribution. Classification model might provide incorrect prediction, which will affect the explanation. If the explainer reflects the difference between the two instances - it is considered to satisfy novelty metric[40].

3.6.3 Sensitivity

Sensitivity metric is about explaining two instances from different classes, but they should differ only in a single feature value. If this feature is reflected in the explanation of these instances - explainer is considered sensitive[41].

3.6.4 Stability

Stability is about explaining similar instances of the same class, using the same classification model. If explaining similar instances produces similar explanations - explainer is considered stable[40].

3.7 Levenshtein distance

There is another metric of comparing explanation - Levenshtein distance. Originally this method is not used to compare strings, not explanations. However, it is used a measure of similarity between explanations. For this purpose explanations are transformed into strings. The Levenshtein distance between two words is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into

the other.[1]. "The greater the Levenshtein distance, the more different the strings are" [42]. When applies to explanations - it will measure the difference between two explanations.

[1] https://en.wikipedia.org/wiki/Levenshtein_distance

4 Solution

4.1 Classifier selection

After calculating the classification score for each of the selected classifiers, each of them had accuracy over 98%, displayed in the Table 2. Classifiers used two classes: benign and malicious. The training set consist of 160 000 samples and the test set is 40 000 samples. All samples are randomly selected. Original distribution of benign and malicious data is 50%-50%, this proportion is the same in both train and test datasets. Accuracy results were high, so precision and recall are calculated to check if model does not overfit [43]. All values were very high. Another experiment is conducted where the classifier's training set has been divided by benign/malicious with different proportions. At the start, a training set was 50% malicious data, 50% benign, but in order to test classifier performance with different distributions, this percentage has been changed several times. The results of this experiment presented in Table 2.

benign / malicious		60 / 40	70 / 30	80 / 20	90 / 10	40 / 60	30 / 70	20 / 80	10 / 90
KNN	Accuracy	0.998	0.9988	0.99840	0.9984	0.99800	0.99840	0.99720	0.99920
	Precision	0.998	0.9988	0.99840	0.9984	0.99800	0.99840	0.99720	0.99920
	Recall	0.998	0.9988	0.99840	0.9984	0.99800	0.99840	0.99720	0.99920
Decision Tree	Accuracy	0.9996	0.9996	1.00000	0.99920	0.99960	0.99960	0.99960	0.99960
	Precision	0.9996	0.9996	1.00000	0.99920	0.99960	0.99960	0.99960	0.99960
	Recall	0.9996	0.9996	1.00000	0.99920	0.99960	0.99960	0.99960	0.99960
SVC	Accuracy	0.9996	0.9988	0.99760	0.99800	0.99920	0.99960	0.99680	0.99880
	Precision	0.9996	0.9988	0.99760	0.99800	0.99920	0.99960	0.99680	0.99880
	Recall	0.9996	0.9988	0.99760	0.99800	0.99920	0.99960	0.99680	0.99880
Logistic Regression	Accuracy	0.9844	0.9868	0.99160	0.99000	0.98960	0.99000	0.99440	0.99840
	Precision	0.9844	0.9868	0.99160	0.99000	0.98960	0.99000	0.99440	0.99840
	Recall	0.9844	0.9868	0.99160	0.99000	0.98960	0.99000	0.99440	0.99840

Table 2. Classifiers stats for different distribution of benign-malicious data.

All models have very good accuracy for this classification task. In order to understand the nature of this accuracy, new tests were conducted. Training set samples were plotted on a 3D graph, which showed samples location in space. Different combinations of three features for plots were tested and plots reviewed by a person. Also, 2D graphs of feature values for benign and malicious data are plotted in order to see the feature difference between these classes. Not only the vast difference between features exists, but some

features of malicious traffic have constant values. To be precise - 26% of features from Mirai traffic has a constant value. This means that classifiers can easily distinguish data by these constant features, which explains a high accuracy. Features with non-constant values also have a high variance of values between benign and malicious classes.

Since all of the selected models have so good scores - accuracy is not a satisfying criteria for classifier selection. The number of classes is changed to eights, and classification models became bigger, which increased interpretation time. As was mentioned before - interpretation is a rather computationally intensive task, which means it is best to pick the smallest model. The eight classes represent the eight types of traffic, used in the database. Thus, further classifiers meant to distinguish traffic types. Decision Tree is selected, because a full version of this model takes only 4.72 seconds to interpret by SHAP. Other models have much higher time, KNN requires 201.50 seconds to explain a single instance. Also, limiting Decision Tree to 20 leaves decreased accuracy to 94%, but also explanation time to 4.01 seconds. Since this study is about explanations - the slight loss in accuracy is acceptable, as now it is possible to explain more instances at the same time period. Thus, a Decision Tree model with 20 leaves is used for all the experiments in this study. LIME explains instances quickly, it takes approximately 0.45 second to explain a single instance. The hardware used for explanations is Intel i7-7700HQ CPU and 16Gb of RAM.

Initial Decision Tree model was relatively simple, but there was an option to simplify classifiers without reducing the number of features - limit the tree itself. After several experiments, it was found that the best way to limit the tree was to limit leaves count. Other options, like limiting the number of nodes or limiting the depth of the tree were producing a slightly worse model in terms of classification. Limiting the number of leaves made a tree more readable for a person. It is possible and simple to manually track a random instance explanation path, if such action is required.

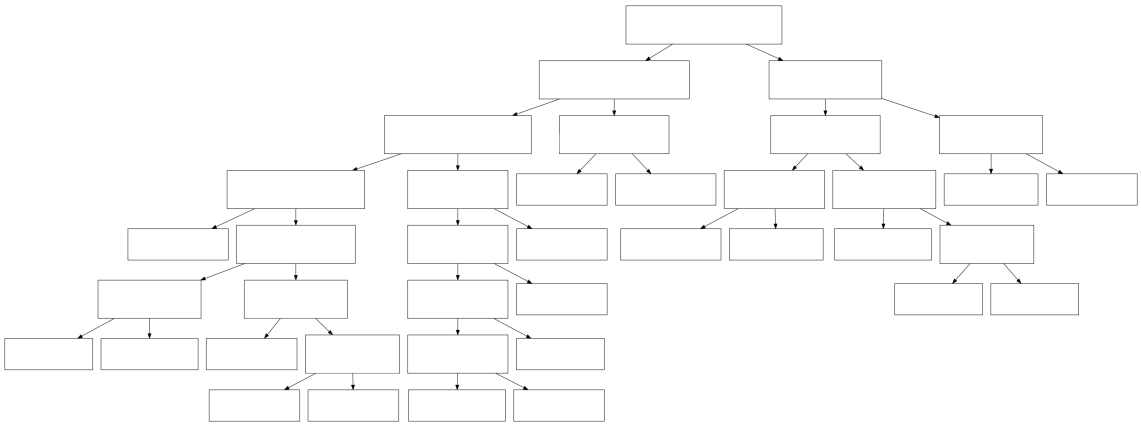


Figure 2. A sample Decision Tree model.

Figure 2 shows a sample Decision Tree model which is relatively small and has 94% prediction accuracy. Only the structure is displayed.

4.2 Feature selection

The main results of feature selection were taken from Fisher score algorithms. Top 10 features selected by Fisher score are:

Feature name	Fisher score
MI_dir_L0.01_weight	1.6701711286616112
H_L0.01_weight	1.6701709777546905
MI_dir_L0.1_weight	1.1319683902655353
H_L0.1_weight	1.1319683445113773
MI_dir_L1_weight	0.9136958345568814
H_L1_weight	0.9136958345216508
MI_dir_L5_weight	0.9041175671598903
H_L5_weight	0.9041175671598903
MI_dir_L3_weight	0.8951098960478836
H_L3_weight	0.8951098960478836

Table 3. Top 10 Fisher score features.

These ten features can be used to distinguish and classify samples between the main eight

classes effectively. For the most complete interpretation results of samples thought the dataset - all 115 features had to be used. In specific cases for comparing interpretations between different machine learning models - these models could be trained on a dataset limited to selected features. These are experiments which compare interpretations of different classification models, for the same instances.

4.3 LIME/SHAP explanation analysis

4.3.1 LIME Feature selection

- "All class" results
  It was mentioned earlier that LIME has explanation results for every class while explaining a single instance. For example, a results of LIME explanation for two classes with five features look like this:

0:	[(9, -0.07845702205219122),	1:	[(9, 0.12172771209813148),
	(24, -0.07729221580720647),		(24, 0.1170737221747649),
	(12, -0.048070758341648805),		(12, 0.07684796957194374),
	(21, -0.042675126997335645),		(27, 0.0714998775720784),
	(6, -0.04151511731223725)],		(6, 0.06206422929370492)]

  Where 0 and 1 are id of class, and a list of tuples are feature weights. For example, the first row contains tuple (9, -0.07845702205219122). 9 is the feature id, and -0.07845702205219122 is the feature weight. Since it is the first in the list - its rank is 1, the highest.
  For all further experiments, the parameters for LIME will remain the same: dataset with all 115 features and top ten features to explain. As presented above, in the example of LIME explanation - for a single instance LIME provides explanations for all classes. In order to use all the computed data for analysis, statistics about feature selection for LIME is divided into two separate results sets.

  1. "Target result" results set - is the results gathered only from the LIME explanation of an original class of the instance.
  2. "All results" results set is built from all LIME class explanations. In the example above, LIME returned to classes explanations for a single instance. While

"Target results" will count only explanations of the instance class, "All results" will take both explanations and extract feature data for current experiment.

Results are presented in the Table 4, where each feature has a percentage which indicates how many times out of all explanations it has been selected for "all class" explanation. These are top five features, selected by LIME.

Feature name	Target result	All results
HH_L3_weight	79.75%	87.18%
H_L0.01_mean	75.88%	75.27%
H_L0.1_weight	64.38%	63.66%
H_L0.01_weight	56.25%	63.19%
MI_dir_L0.01_weight	52.65%	57.78%

Table 4. LIME feature selection comparison.

- "One class" results

  In this experiment, results have been grouped by class and each LIME explanation now belongs to a certain class, same as instance does. Every class has a set of features that is selected for explaining its instances. In the Table 5 presented top five features selected by LIME. Every class has at least one feature, that has been selected for every explanation of that class. These features have 100% selection. Fisher score is included in the table to compare if features with the highest selection have high Fisher score. This would mean, that for explanations LIME selects features that are best for distinguishing dataset classes.

Class name	Feature name	Selection %	Fisher score
ack	H_L0.01_mean	100	0.751917
	MI_dir_L0.01_weight	55	1.670171
	HH_L0.1_magnitude	46	0.258556
	HH_L0.1_mean	35	0.333871
	HpHp_L3_mean	33	0.328307

Class name	Feature name	Selection %	Fisher score
benign	MI_dir_L0.01_weight	100	1.670171
	MI_dir_L0.1_weight	91	1.131968
	HH_L3_weight	90	0.176405
	H_L0.1_weight	88	1.131968
	HpHp_L0.01_weight	83	0.049858
combo	HH_L3_weight	100	0.176405
	MI_dir_L0.1_variance	100	0.462327
	H_L1_mean	100	0.650357
	H_L0.1_weight	96	1.131968
	MI_dir_L1_variance	67	0.545364
junk	H_L1_mean	100	0.650357
	MI_dir_L0.1_variance	100	0.462327
	HH_L3_weight	100	0.176405
	H_L0.01_mean	99	0.751917
	H_L0.1_weight	96	1.131968
scan	H_L0.1_weight	100	1.131968
	HH_L3_weight	100	0.176405
	MI_dir_L0.1_weight	99	1.131968
	H_L0.01_mean	96	0.751917
	MI_dir_L0.01_weight	84	1.670171
syn	H_L1_mean	100	0.650357
	H_L0.01_mean	98	0.751917
	HpHp_L0.01_weight	95	0.049858
	H_L0.1_weight	91	1.131968
	H_L0.01_weight	88	1.670171
tcp	H_L0.01_weight	100	1.670171
	HpHp_L0.01_weight	100	0.049858
	HH_L3_weight	100	0.176405
	H_L0.01_mean	94	0.751917
	HH_jit_L0.01_mean	38	0.537979
udp	H_L0.01_mean	100	0.751917
	H_L0.01_weight	92	1.670171
	HH_L3_weight	60	0.176405
	HH_jit_L5_mean	58	0.484593
	HpHp_L0.01_weight	46	0.049858

Table 5. LIME features for every class.

Statistics about features that has less than 100% selection frequency shows interesting results. After the first features that are selected every time, following features' selection percentage decreases very quickly. The best way it can be displayed - in plots. Figures 3-6 show how feature selection drops with the increase of feature rank. X-axis is the order of a feature in explanation. Y-axis is the percentage of its selection through all explanations in the experiment. For example, "ack" class has a quick drop from 100% to 55% and then 46%. This means, that fewer features required to interpret "ack" classifications, as the only the first rank feature is used in all explanations. On the other hand, "combo" class has top three feature ranks in all explanations. The fourth rank has 96% and the fifth 67%. Which indicates, that first four features explain "combo" class, while the fifth has a big fall in feature selection percentage.

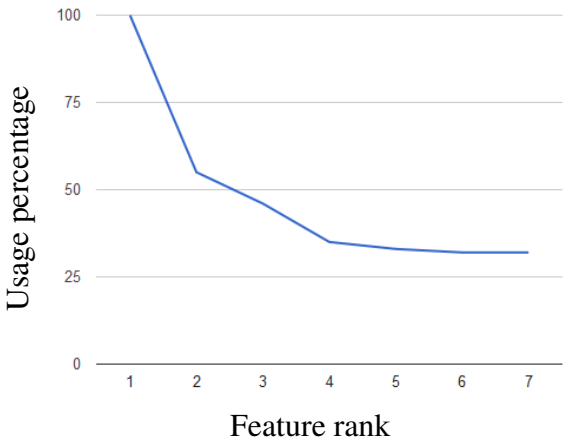


Figure 3. ack feature selection.

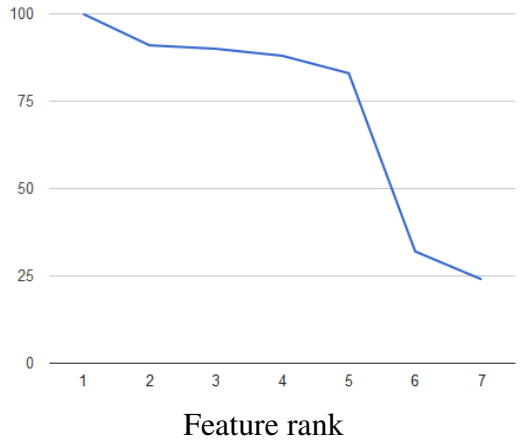


Figure 4. benign feature selection.

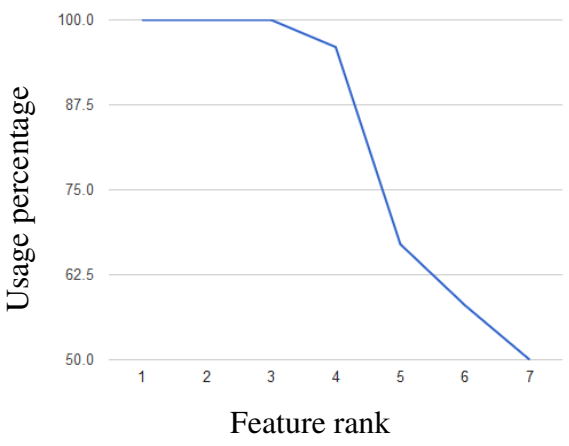


Figure 5. combo feature selection.

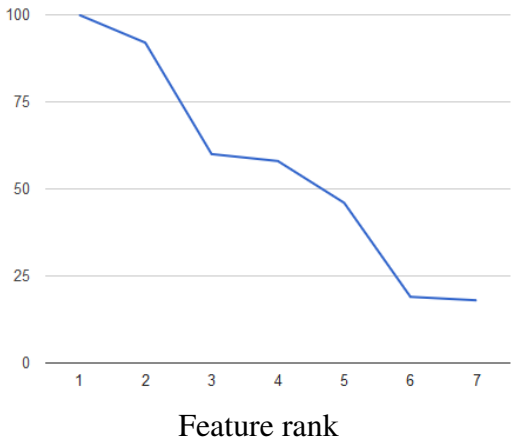


Figure 6. udp feature selection.

4.3.2 SHAP Feature selection

All parameters for SHAP experiments are the same as for LIME. The same dataset with the same samples order. Eights classes, 10 000 instances in each class, feature count is 115. SHAP does not have a "number of features to explain" parameter and sets it itself, which is why in some of the experiments SHAP used less than ten features.

- "All class" results
  Only top 5 features are selected from SHAP explanations for "all class" results. SHAP does not have the explanations for every class, like LIME does. Thus, current results are just the feature selection through SHAP explanations. Results presented in the Table 6.

Feature name	Feature selection
HH_L3_weight	34.38%
HH_jit_L0.01_weight	25.63%
MI_dir_L0.01_weight	25.25%
MI_dir_L0.1_weight	25.13%
H_L0.01_variance	24.38%

Table 6. Top 5 features picked by SHAP.

- "One class" results.

  SHAP selects less than five features for explaining some classes. Classes like "benign", "scan" and "tcp" have only three features. Class "syn" has only two. On the other hand, classes "combo", "junk" and "udp" have more five features and they all are presented in the Table 7. Fisher scores are included as well.

Class name	Feature name	selection %	Fisher score
ack	HH_L0.1_magnitude	100	0.258556
	HH_jit_L5_mean	100	0.484593
	HH_L3_weight	69	0.176405
	HH_jit_L0.01_weight	1	0.302623
benign	HH_L3_weight	100	0.176405
	H_L0.01_weight	46	1.670171
	H_L1_mean	8	0.650357
combo	HH_jit_L0.01_weight	100	0.302623
	HpHp_L0.01_weight	98	0.049858
	MI_dir_L0.01_weight	94	1.670171
	HH_L0.1_weight	94	0.208163
	H_L0.01_variance	93	0.500298
	MI_dir_L1_variance	82	0.545364
	H_L1_weight	26	0.913696
junk	MI_dir_L0.01_weight	100	1.670171
	H_L0.01_variance	99	0.500298
	H_L1_weight	83	0.913696
	HH_L0.1_weight	81	0.208163
	H_L0.01_mean	81	0.751917
	HH_L3_weight	77	0.176405
	MI_dir_L0.1_variance	75	0.462327
scan	MI_dir_L0.1_weight	100	1.131968
	MI_dir_L0.01_weight	7	1.670171
	HpHp_L0.01_weight	6	0.049858
syn	HH_jit_L0.01_weight	100	0.302623
	HH_L3_weight	100	0.176405
tcp	MI_dir_L0.01_weight	100	1.670171
	HH_L0.1_weight	19	0.208163
	HpHp_L0.01_weight	11	0.049858

Class name	Feature name	Selection %	Fisher score
udp	H_L0.01_mean	100	0.751917
	H_L0.01_weight	100	1.670171
	MI_dir_L0.1_weight	100	1.131968
	H_L1_mean	99	0.650357
	H_L0.1_weight	99	1.131968
	MI_dir_L0.1_variance	26	0.462327
	H_L3_variance	9	0.464862

Table 7. SHAP features for explaining every class.

Same as LIME, SHAP has a quick decrease of the feature selection while going along the feature order. When the feature has a lower rank - the number of times it is picked decreases.

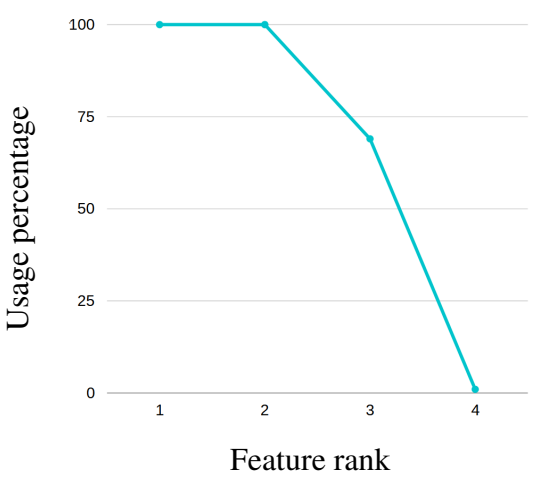


Figure 7. ack feature selection.

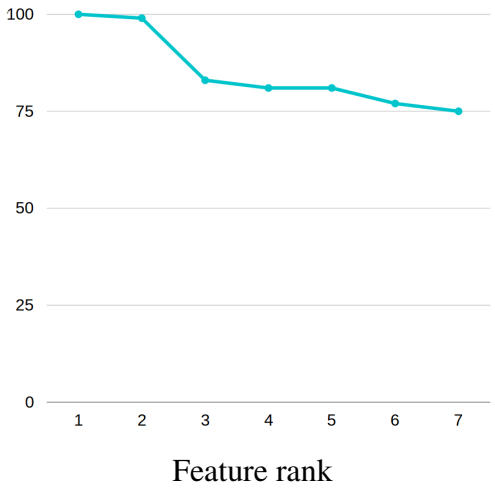


Figure 8. junk feature selection.

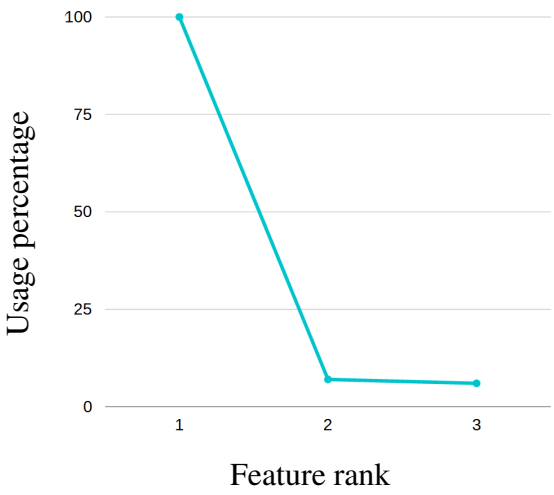


Figure 9. scan feature selection.

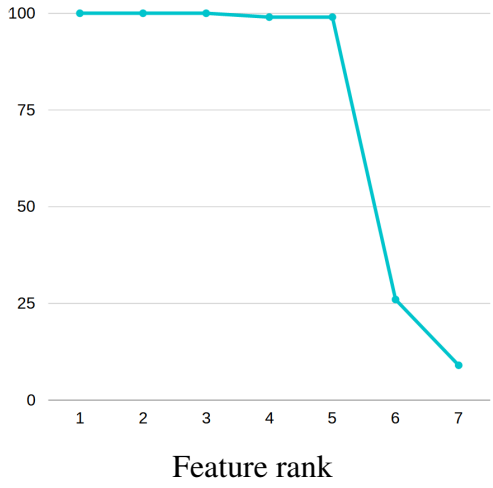


Figure 10. udp feature selection.

Feature selection for some classes is displayed in Figures 7-10. SHAP has a specific feature selection patterns for each class. For example, "junk" class has all features with relatively high selection percentage - 75% and higher, while "scan" class is explained by only three ranks, and the second and third have 7% and 6% feature selection. Which means, that SHAP explain most of "scan" instances with the first rank feature.

Both explainers did not have a single feature, that has been used 100% times for "all class" explanations. However, experiments by class revealed that they both select the same features for explaining specific classes, also known as "one class" explanations. LIME and SHAP have a least one feature that is present in all explanations for a single class. LIME selects a set of features, which it uses for every sample, and the rest of features that are selected have much lower selection frequency. SHAP has similar behavior, but feature selection drops faster for some classes. When LIME has fallen in feature selection from 100% to approximately 50-60%, SHAP has a drop from 100% to approximately 15-20% or less. This means that SHAP uses lower quantity of different features for explanations of classes like "ack", "scan" or "syn". No connection between feature selection and Fisher score is found. Features, selected by LIME and SHAP to explain 100% instances of a particular class does not necessary have the highest Fisher score. For example, LIME selects the feature `H_L0.1_weight` 91% times to explain "syn" class, and it has higher Fisher score than `H_L1_mean`, which is selected 100% times. SHAP selects `MI_dir_L0.01_weight` only 7% for "scan" class, but it has higher Fisher score than `MI_dir_L0.1_weight`, which is selected 100% times. This leads to a conclusion, that LIME and SHAP have different method for distinguishing classes, than Fisher score.

4.3.3 LIME Feature weights

Experiment with gathering information about LIME feature weights consists of several parts:

1. Pick top features from LIME explanations
2. Extract feature weights from the explanations
3. Interpret results
4. Repeat for "all class" and "one class"

"All class" experiment with LIME feature weights involved top five features mentioned in LIME Feature selection. Results are collected in a form of table and histograms. A table contains mean values, standard deviation and a range of values for specific feature. A histogram bars represent a number of samples, explained by LIME with specific feature weight. While table provides an overview of feature weights distribution, the histogram provides details of which weight was used how often. A histogram contains three types of bars: "top 1", "top 2" and "top 3". These names mean that each bar represent feature weights of a certain rank. For example, "top 1" represent feature weights, while this feature has the first rank. When the feature has the second or the third rank inside explanation, its weights fall under "top 2" and "top 3" category respectively.

Table 8 contains results of feature weights experiment. Standard deviation for all features is high compare to mean values. For example, feature `MI_dir_L0.01_weight` has approximately 20 times higher standard deviation than mean value. This means that feature weights are dispersed through the range of value from -0.121343 to 0.089513, and not concentrated around the mean value.

Feature name	Mean	Std	Value range
H_L0.01_mean	0.002044	0.031956	[-0.031558, 0.078971]
HH_L3_weight	0.002823	0.064772	[-0.108169, 0.135268]
MI_dir_L0.01_weight	-0.003846	0.063329	[-0.121343, 0.089513]
H_L0.01_weight	-0.007813	0.035421	[-0.071908, 0.036470]
H_L0.1_weight	-0.001354	0.054600	[-0.116605, 0.052935]

Table 8. Top 5 LIME features weights.

Histograms are built for every class, but only one sample is presented due to high number of images. X-axis is the feature weight, y-axis is the number of explanation, and different bars show the feature weights distribution through the range. As an example, in Figure 11 presented feature MI_dir_L0.01_weight, which was just mentioned in the Table 8, as an example. This visualization shows how feature weights are distributed and helps understand the nature mean and std values for this particular feature.

Figure 11 shows, that there are several groups of explanations using feature MI_dir_L0.01_weight, and each group has its own range of different weight. Blue bars of the left represent explanations where this feature had lower weights and was used at the first rank. Other bars to the right represent explanations with much different weight, which explains the high value of std, compare to mean.

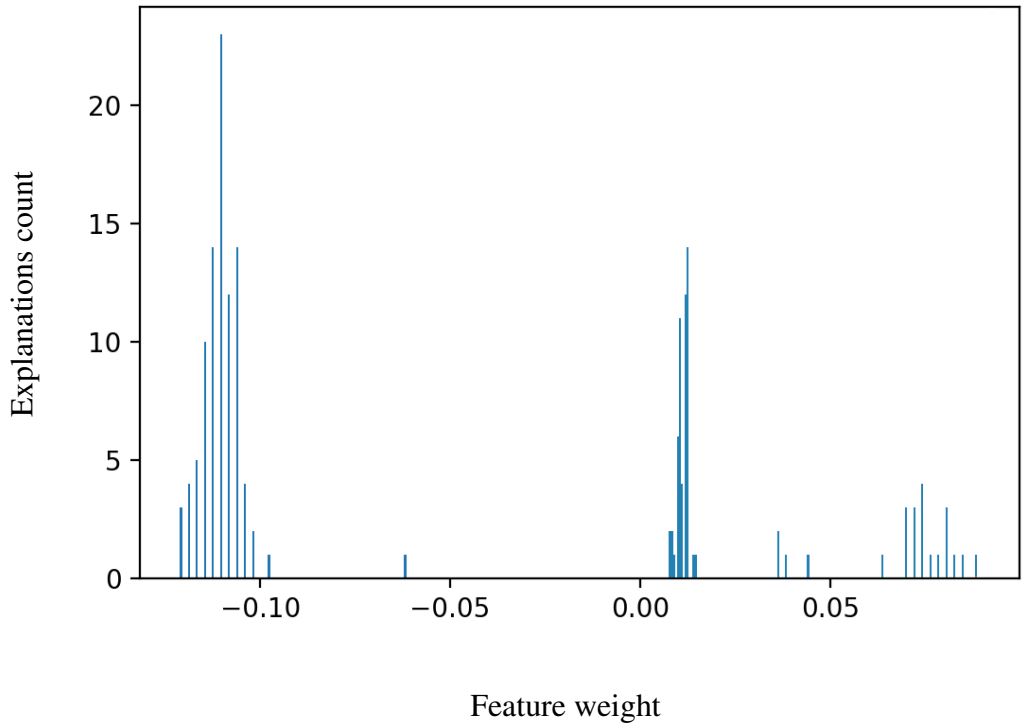


Figure 11. `MI_dir_L0.01_weight` feature weights .

Results about "all class" feature weights for LIME show, that all of the most used features have high standard deviation. This might be related to the nature of current experiment. Random sample were taken from all classes, and different classes might have different weights for the same feature. "One class" experiment will show if the standard deviation is smaller compare to "all class".

"One class" results have the same structure as "all class". They consist of the table with feature weights data and histograms that visualize these numbers, showing weights distribution. Results are being calculated for features, mentioned in LIME Feature selection for "one class" explanations. Results are presented in the Table 9, and features within a single class are sorted based on absolute mean value of feature weight. Weights can be negative, which represent a side to which feature is pushing the classification output. The most influential features in the explanation have the highest absolute weight values.

Class name	Feature name	Selection	Mean	Std	Value range
ack	H_L0.01_mean	100%	0.011946	0.001893	[0.008429, 0.01711]
	MI_dir_L0.01_weight	55%	0.011381	0.001441	[0.008615, 0.014292]
	HH_L0.1_magnitude	46%	0.006375	0.001037	[0.004735, 0.008819]
	HpHp_L3_mean	33%	0.00561	0.001048	[0.001773, 0.007583]
	HH_L0.1_mean	35%	0.005435	0.001377	[-0.001299, 0.006926]
benign	MI_dir_L0.01_weight	99%	-0.103093	0.026648	[-0.121344, -0.0]
	HpHp_L0.01_weight	83%	0.04902	0.003989	[0.038841, 0.057404]
	HH_L3_weight	90%	-0.023933	0.003424	[-0.031715, -0.015561]
	MI_dir_L0.1_weight	91%	-0.02252	0.004285	[-0.033755, -0.012427]
	H_L0.1_weight	88%	-0.014905	0.004006	[-0.02632, -0.007177]
combo	HH_L3_weight	100%	0.122796	0.005623	[0.107129, 0.135269]
	MI_dir_L0.1_variance	100%	-0.037672	0.00451	[-0.050889, -0.029252]
	H_L0.1_weight	96%	0.035491	0.00401	[0.025537, 0.043736]
	H_L1_mean	100%	0.033783	0.003867	[0.025973, 0.044099]
	MI_dir_L1_variance	67%	-0.016444	0.003453	[-0.02661, -0.009902]
junk	H_L0.1_weight	96%	0.044909	0.003644	[0.032129, 0.052935]
	H_L1_mean	100%	0.04419	0.003638	[0.035403, 0.056103]
	HH_L3_weight	100%	0.04061	0.005263	[0.02987, 0.051256]
	MI_dir_L0.1_variance	100%	0.036487	0.003867	[0.02593, 0.048057]
	H_L0.01_mean	99%	-0.02321	0.003519	[-0.031558, -0.013882]
scan	H_L0.1_weight	100%	-0.103595	0.005065	[-0.116606, -0.085588]
	HH_L3_weight	100%	-0.095576	0.004908	[-0.10817, -0.081819]
	MI_dir_L0.01_weight	84%	0.075591	0.005294	[0.063292, 0.089514]
	MI_dir_L0.1_weight	99%	0.020653	0.004445	[0.011158, 0.032983]
	H_L0.01_mean	96%	-0.019509	0.004098	[-0.02779, -0.009011]
syn	H_L1_mean	100%	-0.077897	0.004441	[-0.091125, -0.067054]
	HpHp_L0.01_weight	95%	-0.051892	0.003758	[-0.060472, -0.044547]
	H_L0.1_weight	91%	0.036419	0.003109	[0.028274, 0.043183]
	H_L0.01_weight	88%	0.018911	0.002881	[0.011859, 0.02539]
	H_L0.01_mean	98%	-0.016004	0.003251	[-0.025371, -0.008459]
tcp	H_L0.01_weight	100%	-0.0648	0.002931	[-0.071908, -0.057105]
	HpHp_L0.01_weight	100%	-0.033323	0.002161	[-0.038459, -0.027547]
	HH_L3_weight	100%	-0.011726	0.001976	[-0.015954, -0.007297]
	H_L0.01_mean	94%	-0.009103	0.002084	[-0.014676, -0.004069]
	HH_jit_L0.01_mean	38%	0.002034	0.003001	[-0.004137, 0.006439]

Class name	Feature name	Selection	Mean	Std	Value range
udp	H_L0.01_mean	100%	0.068971	0.004663	[0.054964, 0.078971]
	HpHp_L0.01_weight	46%	-0.044101	0.002609	[-0.049587, -0.037559]
	H_L0.01_weight	92%	-0.024332	0.003551	[-0.032546, -0.014157]
	HH_L3_weight	60%	-0.014559	0.002458	[-0.019259, -0.008337]
	HH_jit_L5_mean	58%	-0.008404	0.002269	[-0.01439, -0.00336]

Table 9. LIME feature weights for every class.

The Table 9 shows that features weights for a single class explanations are much less dispersed compare to the "all class" results. Standard deviation in "one class" feature weights is at least 10 times lower than in "all class" results. For example, let us examine one of the features from the table: feature `H_L0.01_mean` for explaining "ack" class has std = 0.001893, while in "all class" results its std was 0.031956. Another example, feature `H_L0.01_weight` with std = 0.054600 in "all class" explanations, has std = 0.002931 for explaining "tcp" class. Figure 12 visualizes feature weights distribution in "tcp" explanations. This means, that feature weights does not differ very much within explanations of samples from the same class. And since feature weights is the most important part of the explanation - this leads to a conclusion, that LIME explanations of the same features are similar within one class.

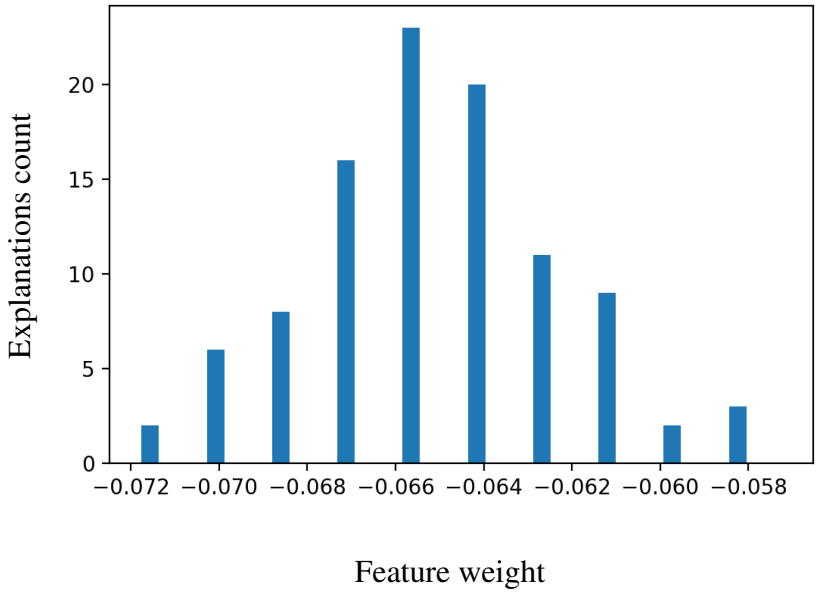


Figure 12. `H_L0.01_weight` weights sample within tcp class.

As a conclusion - the features used by LIME to explain random samples through the dataset can have various weights. On the other hand, features for explaining instances of specific classes have more similar weights, more equally distributed through the range of feature weights values. This mean, that LIME consistently explain instances of the same class with the same feature and the same feature weight.

4.3.4 SHAP Feature weights

The solution to experiments with SHAP results is practically the same as LIME experiments. The only difference is that for SHAP, "all class" experiments are not conducted. In SHAP Feature selection section, "all class" experiment for SHAP showed that SHAP has very low feature selection for top five features, of maximum 34.38% and lower. On the other hand - "one class" SHAP experiment results are similar to the LIME results. SHAP has a feature, that is always used to explain every class. Since analysis of "all class" features with such low selection would not bring useful results - only "one class" feature weights are analyzed.

Results are collected and presented in the same way as for LIME. A table with feature weights data and histograms to visualize the weights distribution. Table 10 contains SHAP feature weights data for each feature, mentioned in SHAP Feature selection section for "one class" explanation.

Class name	Feature name	Selection	Mean	Std	Value range
ack	HH_jit_L0.01_weight	1%	0.706184	0.0	[0.706184, 0.706184]
	HH_L3_weight	69%	0.010278	0.465264	[-0.05, 3.846807]
	HH_jit_L5_mean	100%	-0.007405	1.619219	[-2.423188, 1.085]
	HH_L0.1_magnitude	100%	-0.006748	1.596621	[-1.090043, 2.373188]
benign	MI_dir_L0.01_weight	100%	-0.071443	0.440261	[-0.511283, 0.706184]
	HH_L0.1_weight	19%	0.008389	0.129835	[-0.43274, 0.129241]
	HpHp_L0.01_weight	11%	-0.002611	0.992341	[-0.93294, 0.816729]
combo	H_L1_weight	26%	0.006579	0.073155	[-0.084593, 0.295278]
	HH_L0.1_weight	94%	-0.0038	0.111854	[-0.386689, 0.051642]
	MI_dir_L1_variance	82%	0.001451	0.086204	[-0.253193, 0.06105]
	HH_jit_L0.01_weight	100%	-0.001407	0.418	[-0.4558, 0.511372]
	MI_dir_L0.01_weight	94%	-0.00108	0.118486	[-0.200467, 0.158275]
junk	HH_L3_weight	77%	0.001913	0.047417	[-0.259534, 0.020946]
	MI_dir_L0.1_variance	75%	0.000655	0.083981	[-0.023595, 0.359342]
	H_L1_weight	83%	-0.000587	0.104621	[-0.236393, 0.253788]
	MI_dir_L0.01_weight	100%	0.000185	0.157519	[-0.101505, 0.329496]
	H_L0.01_variance	99%	-8.2e-05	0.222097	[-0.462254, 0.142532]
scan	MI_dir_L0.1_weight	100%	-1.6245249	0.292481	[-2.91, 0.03]
	MI_dir_L0.01_weight	7%	0.003912	0.051962	[-0.06, 0.045]
	HpHp_L0.01_weight	6%	0.001449	0.042426	[-0.03, 0.06]
syn	HH_jit_L0.01_weight	100%	6.8e-05	0.210462	[-0.461225, 0.65482]
	HH_L3_weight	100%	-6.8e-05	0.325007	[-1.485957, 0.140942]
tcp	HH_L3_weight	100%	0.0277183	0.410629	[-1.227391, 3.846807]
	H_L0.01_weight	46%	0.00298	0.074301	[-0.182363, 0.110492]
	H_L1_mean	8%	-0.000183	0.083291	[-0.113287, 0.189017]
udp	MI_dir_L0.1_variance	26%	-0.006918	0.040148	[-0.077389, 0.060504]
	H_L0.01_mean	100%	-0.001575	1.201042	[-1.51413, 1.027618]
	MI_dir_L0.1_weight	100%	0.001142	0.123386	[-0.217381, 0.166731]
	H_L0.01_weight	100%	0.000808	0.882964	[-0.818789, 1.14688]
	H_L3_variance	9%	-0.000782	0.032182	[-0.044359, 0.031918]

Table 10. SHAP feature weights for every class.

Standard deviation for every class is very high compare to LIME results. For example, class "ack" has feature HH_L0.1_magnitude with std = 1.596621. In Figure 13 it is displayed, that explanations with this feature are divided into two parts. In the first part SHAP assigned this feature with the weight of -1.090043. In the second part,

HH_L0.1_magnitude has the weight of 2.373188. These are the only two major selections of this feature in explanations by SHAP, hence the std is high.

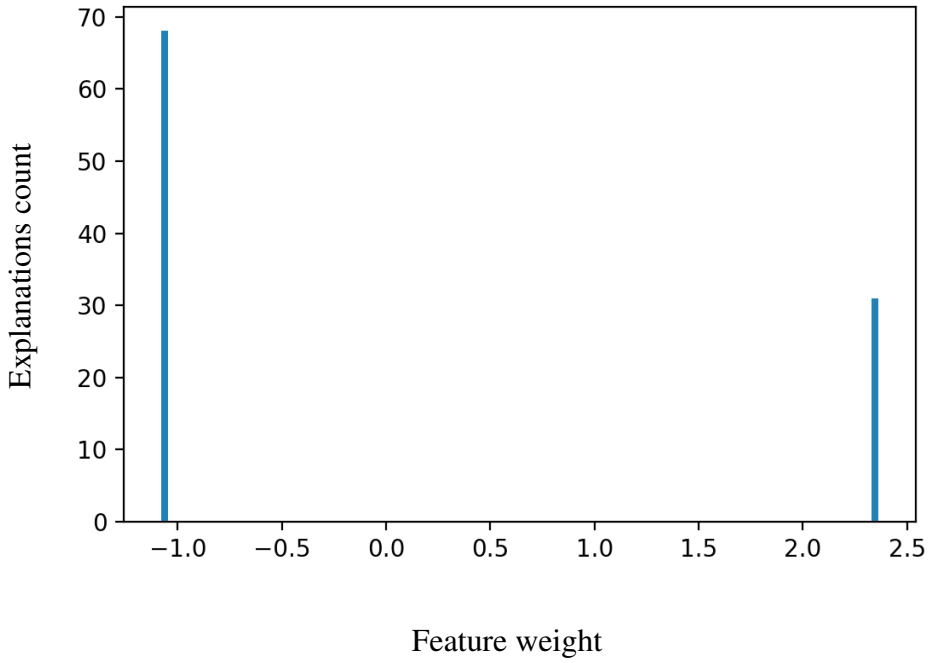


Figure 13. HH_L0.1_magnitude weights sample for "ack" class .

Another example - feature MI_dir_L0.1_weight for "scan" class. Details of feature weights and their selections in various explanations displayed in the Figure 14. The most number of explanations involving current feature has the same feature weight. A few explanations, presented by the little bar on the left is an exception.

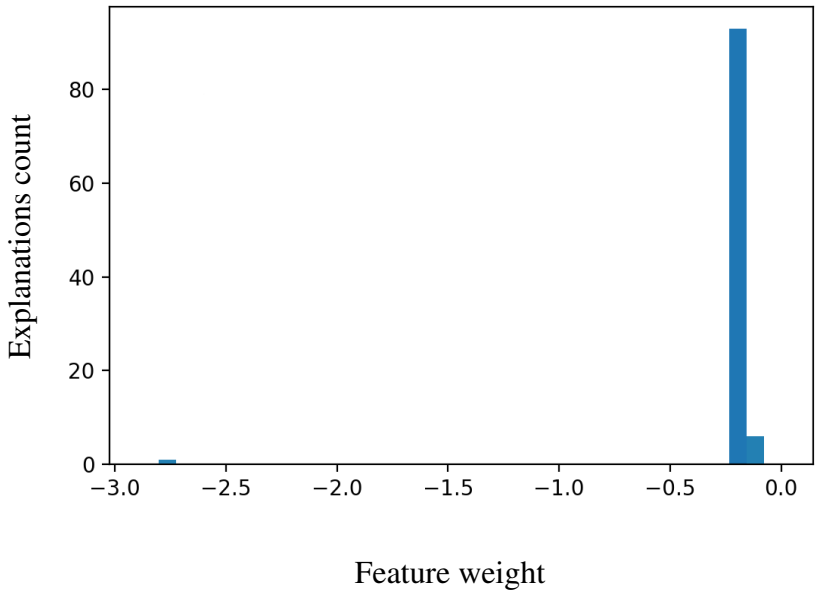


Figure 14. `MI_dir_L0.1_weight` weights sample for "scan" class .

SHAP feature weights experiment showed, that features can be assigned a very different weight, even it is used to explain the samples of the same class. SHAP assigns similar weight to the feature, while the feature rank remains the same through various explanations of the same class. However, if the rank of the feature inside explanation is changes - its weight may vary significantly.

4.3.5 Comparing LIME and SHAP feature weights

In the previous chapter LIME and SHAP feature weights were analyzed separately. Two types of experiments for each method were conducted - "all class" explanations feature weights and "one class" explanations. For "all class" explanations both methods have very different feature weights, which is explained by the different classes explanations for every feature. For "one class" explanations LIME had more similar weights for a single class, while SHAP had less similar weights. However, SHAP weights had less variance within a rank.

For comparing LIME and SHAP, the same two types of experiments are conducted. For "all class" explanations two experiments are conducted to compare LIME and SHAP feature weights. The first one contains LIME top features, and for them - LIME and SHAP feature weights data side by side. Table 11 contains results of this experiment, which provides an overview of difference in feature weights assignment by two interpretability methods.

Feature name	Method	Mean	Std	Value range
HH_L3_weight	LIME	0.002127	0.306949	[-1.485957, 3.846807]
	SHAP	0.002044	0.031956	[-0.031558, 0.078971]
H_L0.01_mean	LIME	0.002520	0.864349	[-1.514130, 1.027618]
	SHAP	0.002823	0.064772	[-0.108169, 0.135268]
H_L0.1_weight	LIME	0.003727	0.095040	[-0.154182, 0.130368]
	SHAP	-0.003846	0.063329	[-0.121343, 0.089513]
H_L0.01_weight	LIME	0.000825	0.870011	[-0.818789, 1.146880]
	SHAP	-0.007813	0.035421	[-0.071908, 0.036470]
MI_dir_L0.01_weight	LIME	-0.000122	0.137576	[-0.200467, 0.329496]
	SHAP	-0.001354	0.054600	[-0.116605, 0.052935]

Table 11. Top 5 LIME "all class" explanations features compare.

The same experiment is conducted for top five features, used by SHAP in "all class" explanations. Results are presented in the Table 12.

Feature name	Method	Mean	Std	Value range
HH_L3_weight	LIME	0.002823	0.064772	[-0.108169, 0.135269]
	SHAP	0.002127	0.306949	[-1.485956, 3.846807]
HH_jit_L0.01_weight	LIME	0.003091	0.012099	[-0.019955, 0.022721]
	SHAP	0.002676	0.330568	[-0.461225, 0.706184]
MI_dir_L0.01_weight	LIME	-0.003847	0.063329	[-0.121344, 0.089514]
	SHAP	-0.000122	0.137576	[-0.200467, 0.329496]
MI_dir_L0.1_weight	LIME	0.000301	0.018635	[-0.033755, 0.032983]
	SHAP	0.000412	0.223918	[-2.910000, 0.166731]
H_L0.01_variance	LIME	0.002353	0.012426	[-0.021661, 0.022276]
	SHAP	-0.000598	0.207733	[-0.462254, 0.390308]

Table 12. Top 5 SHAP "all class" explanations features compare.

These two experiment results demonstrate the differences in the weight value ranges. For the next "one class" experiments this is a problem, as it is difficult to compare mean and std of values from different value distributions. For example, in the Table 12, feature HH_L3_weight has very similar mean values for LIME and SHAP: 0.002823 and 0.002127. But the range of all possible weight for each method varied greatly: [-0.108169, 0.135269] for LIME and [-1.485956, 3.846807] for SHAP.

In order to compare feature weights for "one class" explanation, weights have been rescaled to a range from -1 to 1. Next experiments compare LIME and SHAP weights for every class. Some weights are missing, as interpretation methods do not always use the same feature for a specific class explanation. Thus, in such cases only the method which has used a particular features is listed, and the other one is skipped. The first experiment is about top LIME features for explaining every class and SHAP weights for these features. Results presented in the Table 13, weights are rescaled to [-1, 1].

Class name	Feature name	Method	Mean	Std
ack	H_L0.01_mean	LIME	-0.18967403	0.43621635
	HpHp_L3_mean	LIME	0.32081254	0.36080531
	HH_L0.1_magnitude	LIME	-0.19697339	0.50785627
		SHAP	-0.37440232	0.92204101
	HH_L0.1_mean	LIME	0.63741932	0.33480289
	MI_dir_L0.01_weight	LIME	-0.02541884	0.5075023
benign	MI_dir_L0.01_weight	LIME	-0.6991807	0.43921612
	HpHp_L0.01_weight	LIME	0.0967261	0.42974199
		SHAP	-0.00261143	0.99234192
	HH_L3_weight	LIME	-0.0365644	0.42386391
	MI_dir_L0.1_weight	LIME	0.05354771	0.40179158
	H_L0.1_weight	LIME	0.19258562	0.41850227
combo	HH_L3_weight	LIME	0.11354598	0.39960775
		SHAP	-0.21985252	0.41731595
	MI_dir_L0.1_variance	LIME	0.22172156	0.41684135
		SHAP	-0.39101956	0.54283285
	H_L0.1_weight	LIME	0.09387159	0.4406881
	H_L1_mean	LIME	-0.13823738	0.42674245
		SHAP	-0.16023687	0.84735965
	MI_dir_L1_variance	LIME	0.21698162	0.41330359
		SHAP	0.62068203	0.54864536
junk	H_L1_mean	LIME	-0.15096954	0.35151938
		SHAP	-0.29477908	0.78422856
	H_L0.1_weight	LIME	0.22848561	0.35030844
		SHAP	-0.50654283	0.58508111
	MI_dir_L0.1_variance	LIME	-0.04577232	0.34955856
		SHAP	-0.87334386	0.4386131
	HH_L3_weight	LIME	0.00440196	0.49221714
		SHAP	0.86428221	0.33811408
	H_L0.01_mean	LIME	-0.05546914	0.39812988
		SHAP	-0.40716122	0.33681302

Class name	Feature name	Method	Mean	Std
scan	H_L0.1_weight	LIME	-0.1610608	0.32657657
	HH_L3_weight	LIME	-0.04409836	0.37249468
	MI_dir_L0.01_weight	LIME	-0.06195428	0.40378657
		SHAP	0.14285714	0.98974332
	H_L0.01_mean	LIME	-0.11810917	0.43644021
	MI_dir_L0.1_weight	LIME	-0.12991022	0.40731981
		SHAP	0.97959184	0.19896651
syn	H_L1_mean	LIME	-0.06195428	0.40378657
		SHAP	-0.2349874	0.6921158
	H_L0.1_weight	LIME	0.0926347	0.41710584
	H_L0.01_weight	LIME	-0.06195428	0.40378657
		SHAP	0.3934752	0.43665672
	H_L0.01_mean	LIME	0.10768595	0.38448248
	HpHp_L0.01_weight	LIME	0.07758667	0.47200932
tcp	H_L0.01_weight	LIME	-0.03965918	0.39598687
	HpHp_L0.01_weight	LIME	-0.05860687	0.39600944
	HH_L3_weight	LIME	-0.02330244	0.45656836
	H_L0.01_mean	LIME	0.05082059	0.39296857
	HH_jit_L0.01_mean	LIME	0.16685273	0.56742031
udp	H_L0.01_mean	LIME	0.16690021	0.38849355
		SHAP	0.19016936	0.94505238
	H_L0.01_weight	LIME	-0.10662546	0.38622396
		SHAP	-0.16608854	0.89838525
	HH_L3_weight	LIME	-0.13934182	0.45012604
		SHAP	0.2575248	0.89403843
	HH_jit_L5_mean	LIME	0.0854724	0.4114702
		SHAP	-0.03266409	0.70238397
	HpHp_L0.01_weight	LIME	-0.08777478	0.43376019

Table 13. Comparing LIME features for "one class" explanations.

When weights are rescaled to the same range, some features like H_L0.01_mean and H_L1_mean have similar mean values for LIME and SHAP, but not all of them do. The second experiment will compare SHAP weights for "one class" explanations with LIME weights. Results in the Table 14.

Class name	Feature name	Method	Mean	Std
ack	HH_L0.1_magnitude	LIME	-0.19697339	0.50785627
		SHAP	-0.37440232	0.92204101
	HH_L3_weight	LIME	0.09934655	0.53790342
		SHAP	-0.96906308	0.23879244
	HH_jit_L5_mean	LIME	-0.15322304	0.468722
		SHAP	0.37722545	0.9231082
	HH_jit_L0.01_weight	SHAP	-0.06184096	0.78364923
benign	HpHp_L0.01_weight	LIME	-0.05860687	0.39600944
		SHAP	-0.3784743	0.9422957
	HH_L0.1_weight	LIME	-0.08647955	0.71963325
		SHAP	0.7843832	0.33902564
	MI_dir_L0.01_weight	LIME	-0.699181	0.411483
		SHAP	-0.5806301	-0.218423
combo	MI_dir_L0.01_weight	LIME	0.0978497	0.38964512
		SHAP	0.11158699	0.6605627
	H_L1_weight	LIME	-0.47125637	0.44418859
		SHAP	-0.51998705	0.38515614
	HH_jit_L0.01_weight	LIME	-0.34249122	0.40134431
		SHAP	-0.06036738	0.86437645
	MI_dir_L1_variance	LIME	0.21698162	0.41330359
		SHAP	0.62068203	0.54864536
	HH_L0.1_weight	LIME	-0.02308198	0.31691731
		SHAP	0.74703427	0.51036487
junk	MI_dir_L0.1_variance	LIME	-0.04577232	0.34955856
		SHAP	-0.87334386	0.4386131
	HH_L3_weight	LIME	0.00440196	0.49221714
		SHAP	0.86428221	0.33811408
	MI_dir_L0.01_weight	LIME	-0.01594397	0.32913888
		SHAP	-0.52812149	0.73094498
	H_L0.01_variance	LIME	0.12019034	0.40743675
		SHAP	0.52838084	0.73446289
	H_L1_weight	LIME	0.50393794	0.38034928
		SHAP	-0.03788219	0.42686549

Class name	Feature name	Method	Mean	Std
scan	HpHp_L0.01_weight	LIME	-0.12844803	0.35861406
		SHAP	-0.33333333	0.94280904
	MI_dir_L0.01_weight	LIME	-0.06195428	0.40378657
		SHAP	0.14285714	0.98974332
	MI_dir_L0.1_weight	LIME	-0.12991022	0.40731981
		SHAP	0.97959184	0.19896651
syn	HH_jit_L0.01_weight	LIME	-0.08265201	0.42068349
		SHAP	-0.17334233	0.37715647
	HH_L3_weight	LIME	-0.06524028	0.43163526
		SHAP	0.82665148	0.39954216
tcp	HH_L3_weight	LIME	-0.023302	0.456568
		SHAP	-0.7526494	-0.6771862
	H_L0.01_weight	LIME	0.31132344	0.47675424
		SHAP	-0.16932421	0.89831184
	H_L1_mean	LIME	0.03533347	0.74316546
		SHAP	-0.16119942	0.79181363
udp	H_L0.01_mean	LIME	0.16690021	0.38849355
		SHAP	0.19016936	0.94505238
	H_L0.01_weight	LIME	-0.10662546	0.38622396
		SHAP	-0.16608854	0.89838525
	MI_dir_L0.1_weight	LIME	-0.14800808	0.66794417
		SHAP	0.13780881	0.64244646
	H_L3_variance	LIME	0.04135126	0.74185827
		SHAP	0.14260438	0.84382661
	MI_dir_L0.1_variance	LIME	0.36544234	0.60286534
		SHAP	0.0221157	0.58230495

Table 14. Comparing SHAP features for "one class" explanations.

Results of these experiments show no connection between LIME and SHAP feature weights assignment. These methods use different features to explain the same input instances. In some cases, selected features overlap, however the weights are not similar. Histograms are built in this experiment, in order to visualize feature weights for LIME and SHAP side by side. In the histogram, the weights are not rescaled and used original values. X-axis represent feature weights, y-axis represent explanations count using a

particular weight. Each histogram contains two types of bars - LIME and SHAP. As an example, a few histograms presented in:

- Figure 15: `HH_L3_weight` used by LIME to explain "junk" class
- Figure 16: `H_L0.01_mean` used by SHAP to explain "udp" class

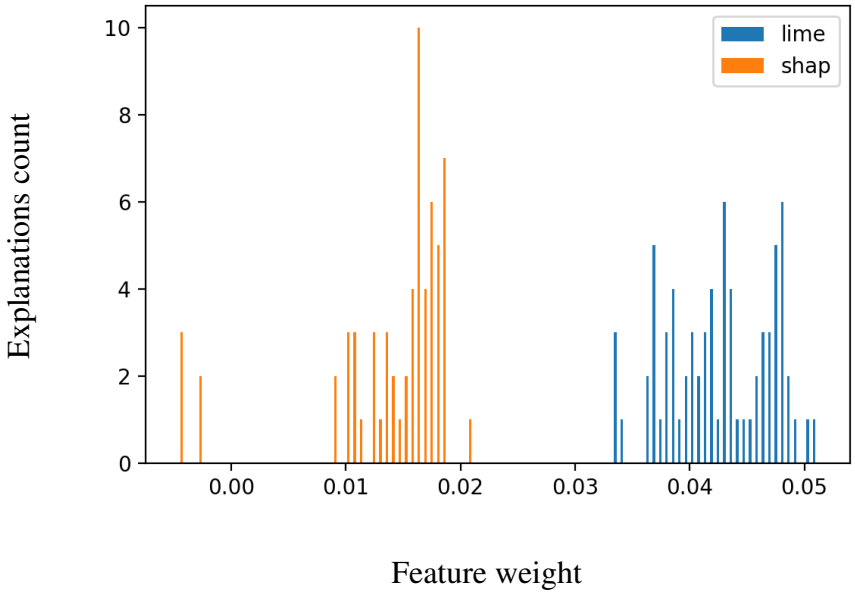


Figure 15. `HH_L3_weight` for "junk" class .

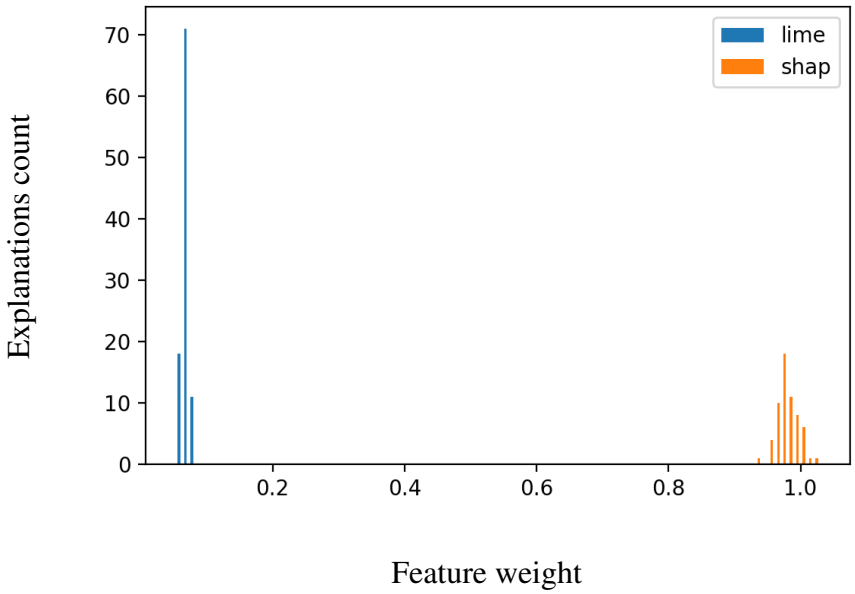


Figure 16. `H_L0.01_mean` for "udp" class .

4.3.6 Explaining points close to decision boundary

The previous experiments were are about explaining random points from the dataset and points from the specific class. However, points that are close to a decision boundary, especially when there are eight classes deserve a separate experiment. The next experiment objective is to find classes with a groups of instances far from the main distribution. In most cases, class samples are grouped and located in a specific area, but classes like "udp", "syn" and "combo" have wider distribution. "Udp" distribution has the Gafgyt udp flooding traffic, which is far from the rest "udp" class. These samples within "udp" class are often incorrectly classified and are displayed in the Figure 17. Red area in the bottom represent concentration of Gafgyt udp traffic instances. Features, used for the plot are picked from the set of most used features in "udp" explanations.

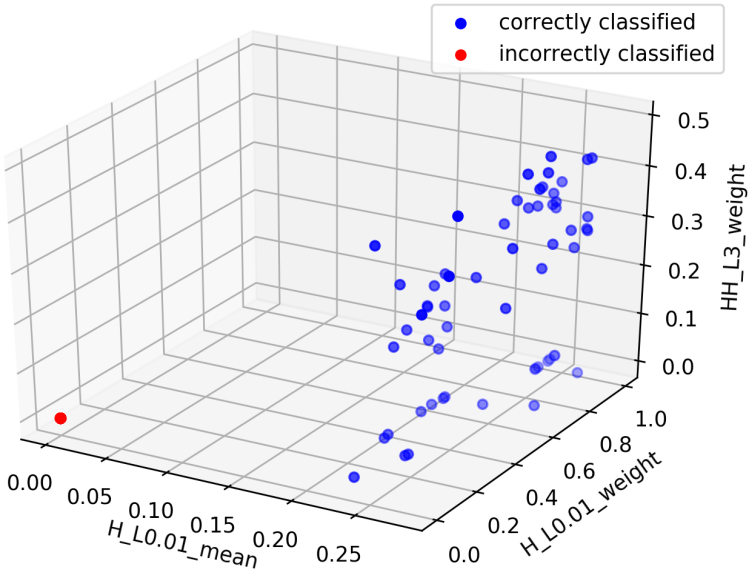


Figure 17. "udp" data points location .

For other classes: "syn" and "combo", there is a solution in the form of a histogram which shows prediction probability for all sample of a single class. In the Figure 18 - a sample graph for "combo" class, with the mean value of prediction probability specified under the graph. Prediction probability is provided after classification, by the machine learning model.

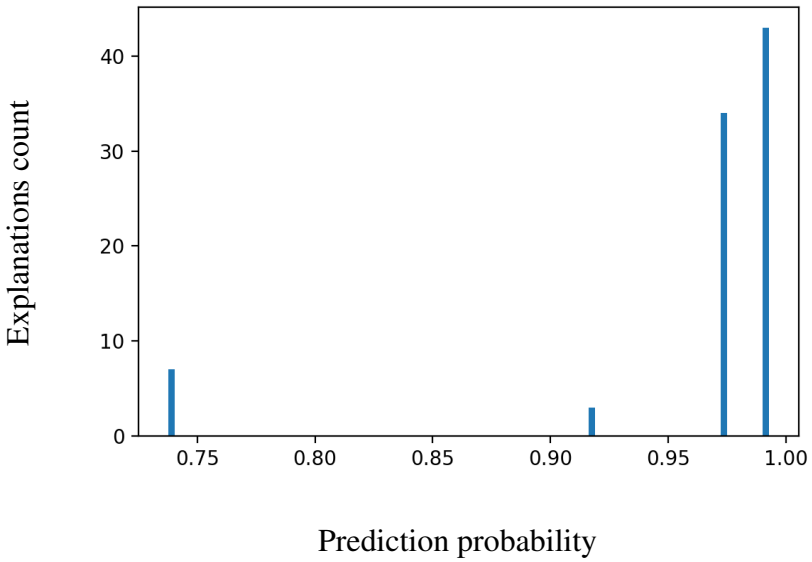


Figure 18. "combo" samples prediction probability .

There is a set of a few different chunks of samples that have the same prediction probability. This means that data is divided into a few sets and there is a set further away from the main distribution. This is the set with a lower prediction probability. A plot of samples from "combo" class is built to confirm this hypothesis. Feature, that are used the most to explain "combo" class are used to plot the data. Samples are colored by their prediction probability. For "combo" class, the set of samples with the lowest prediction probability is displayed in the Figure 19 and is colored with green color. These points are closer to a decision boundary. Other points have prediction probability approximately 93%(see Figure 18), and more than 95%.

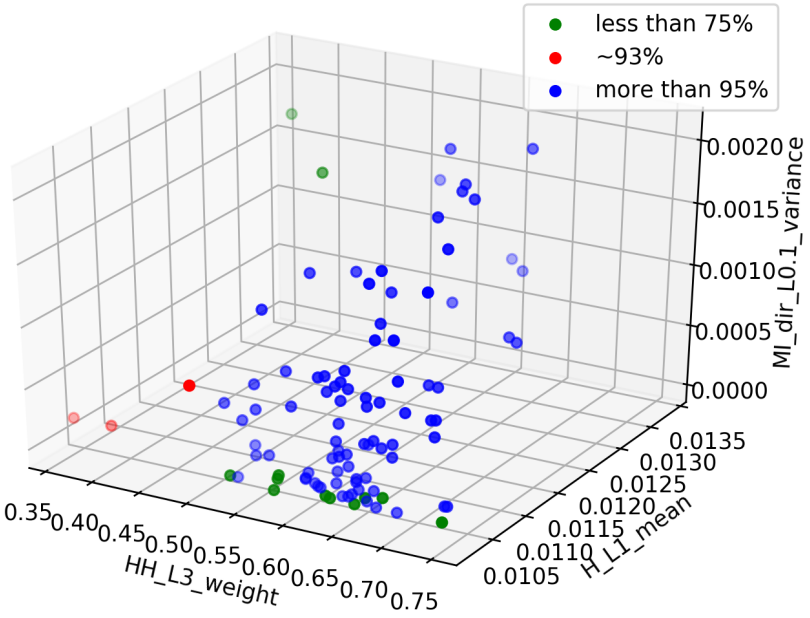


Figure 19. Combo samples location.

Since explanation itself is a set of feature weights - a solution to this experiment is built in a form of average explanation. This is done in order to display an average explanation weight for selected samples. Samples are selected based on the prediction probability from previously presented results. Graphs with mean feature weights give a good overview of LIME and SHAP explanations of points which are closer and further away from decision boundary. Feature rank is set based on the absolute value of mean weight.

The results for this experiment are presented in two graphs for each explainer. One graph shows the mean values of points far from the decision boundary. Another shows the closer one.

- LIME explanations

  On the Figure 20 and 21 "combo" class is presented to demonstrate mean feature values of points close and far from decision boundary.

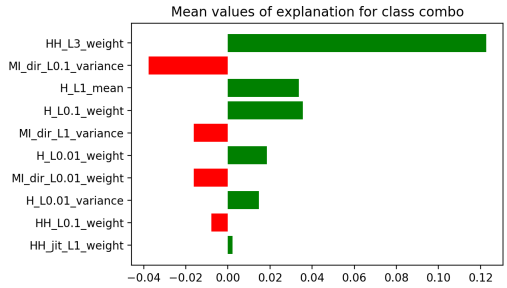


Figure 20. Close to decision boundary.

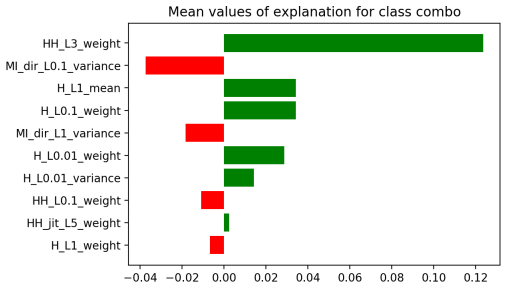


Figure 21. Far from decision boundary.

The two explanations are very similar. The feature order and feature weights are similar. The order of features in explanations differs for the lower ranks, which have much lower weight in the explanation. Features with the highest weights have the same rank and similar weights. As another example, in Figures 22 and 23 presented "udp" explanations.

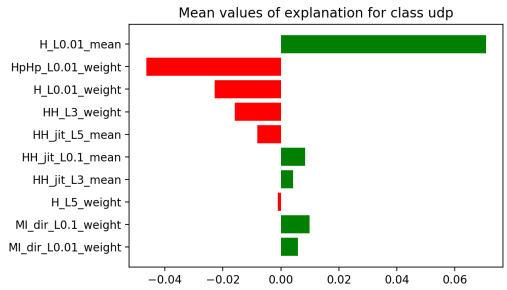


Figure 22. Close to decision boundary.

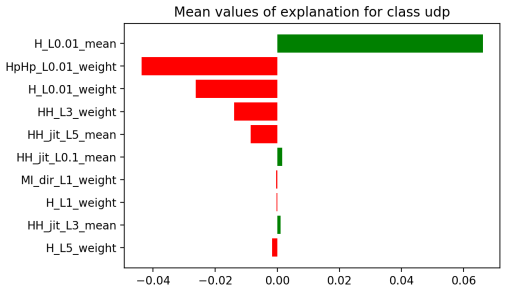


Figure 23. Far from decision boundary.

- SHAP explanations

  SHAP results are presented in the Figures 24 and 25, and the explanations are different. Feature weights are opposite for points that are closer and further away from decision boundary. Nevertheless, the order of features does not vary very much. Some features are missing, but the most important features are on the same order in both cases. This shows that SHAP recognizes points of the same class and most important features for this class, but the weights are different.

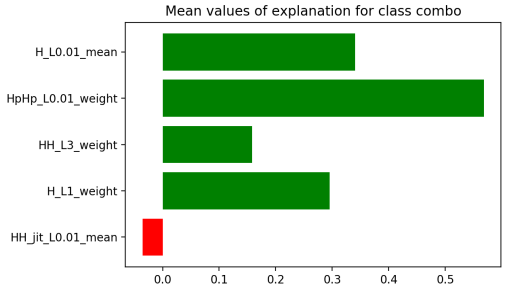


Figure 24. Close to decision boundary.

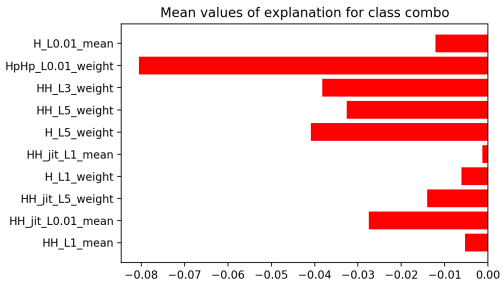


Figure 25. Far from decision boundary.

SHAP "udp" class results presented in Figures 26 and 27. Results conclusion is the same as for "combo" class, and both explanations are very different.

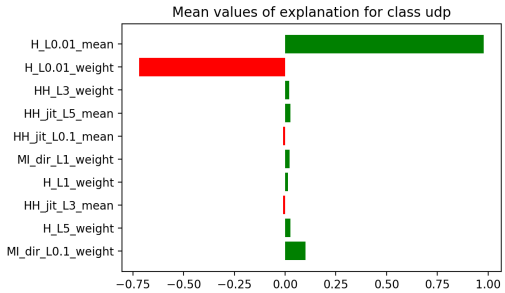


Figure 26. Close to decision boundary.

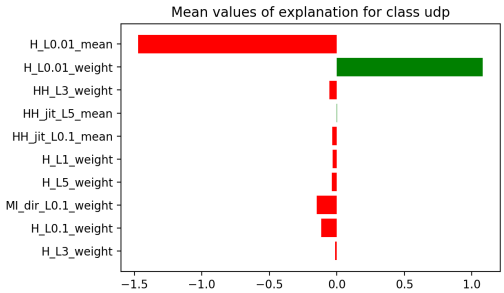


Figure 27. Far from decision boundary.

- Comparing LIME and SHAP explanations

  LIME and SHAP explanations differ from one another, but this does not mean that one is better than another. After all, these points are further away from the main distribution and have higher probability to belong to a different class. The fact that LIME explanation of them is similar to the ones which are far from decision boundary is neither good nor bad. On the one hand, it is good that LIME is consistent with its explanations, on the other - since points are far away, explanation might vary like in SHAP. The variance in feature order is minimal for both LIME and SHAP. But SHAP has a fewer number of features in explanation, that might mean less informative explanation. LIME always explains with ten features, which is not too many for human comprehension. While SHAP may use three or four features, which might prove too little for some cases. In a real-life scenario, it is more likely that there will be fewer points closer to decision boundary than in main class distribution. In such case since SHAP explanations will have fewer features in explanation - it may not be enough to interpret the result, but LIME will always provide as many features as requested, and thus LIME explanation will be more informative.

LIME and SHAP behave differently while explaining points that are close to the decision boundary. LIME explanations remain consistent, feature weights, feature count and feature order are practically identical. With SHAP, it is a little different. Two explanations may contain a different number of features, as presented in Figures 24 and 25. Those features that are present partially have the same order, but weights values differs in up to 6 times. In all cases, the difference between weights is so high, that feature has an opposite sign of influence on the classifier.

4.4 LIME and SHAP and Decision Tree comparison

In Decision Tree, there are many different branches that lead to classes. The model used in experiments contains 18 such branches. The initial parameters for building the model were to limit it to 20 leaves. Thus many possible outcomes of decision path are possible. Approximately 116 thousand for current model. After analysis that there are not many varieties of features count in explanation path, it is decided to have histograms and a separate bar for each possible number of features in the explanation path. Later separate bars for LIME and SHAP are added. The main objective of the next experiment is to compare feature order of LIME and SHAP with the Decision Tree. For this experiment only the data about features that are present in all three explanations are collected. To be precise, 16 features have been used in explanations, and graphs are plotted for each of them, where the x-axis shows feature order in explanations and y-axis is the total number of times this feature has been used with a specific order. Feature order for Decision Tree is the index of the feature in explanation path. The data for this experiment gathered from "all class" explanations. A sample graph is below and it is `MI_dir_L0.1_weight` feature.

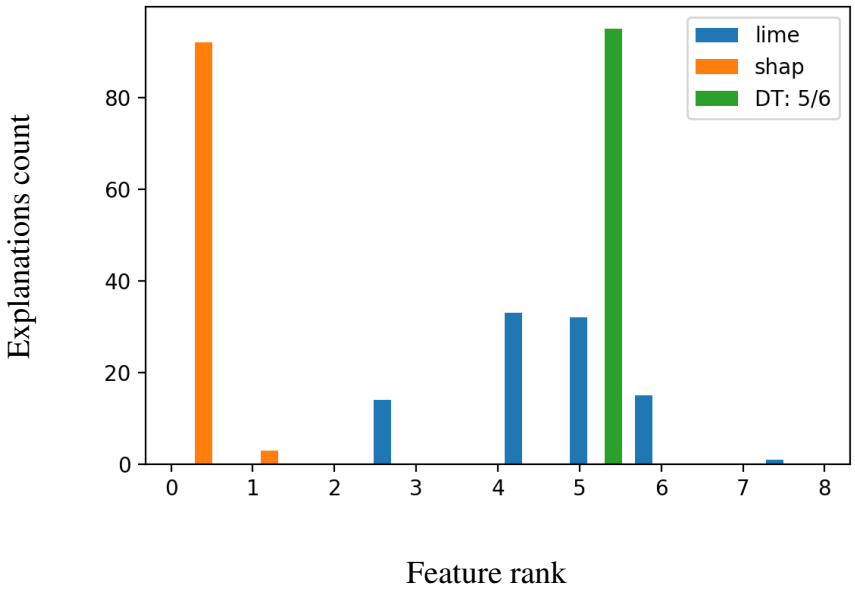


Figure 28. `MI_dir_L0.1_weight` feature order-selection .

In Figure 28, the "lime" bars represent LIME explanations with a certain feature rank, "shap" bars represent SHAP explanations, and "DT: x/y" represent Decision Tree. X is the current feature rank in explanation path, and y is the maximum number of features in explanation path. Multiple "DT" types of bars represent multiple options of decision path.

Results of analyzing feature selection and order are more informative for highly frequently used features. These features are `H_L0.01_mean`, `HH_L3_weight`, `MI_dir_L0.1_weight` and `H_L0.01_weight`. All these features are in the top three most used features to explain different classes by LIME, SHAP and Decision Tree.

For example, feature `HH_L3_weight` is most popular among specified and is on the top of the Decision Tree model. Its order in explanation path is always 0, meaning it is always the first and most significant for classification. On the graph, one can see that is has been used in the top 10 explanation features by LIME and SHAP as well. In a few cases, it was used by SHAP with some higher orders, but these amounts are tiny.

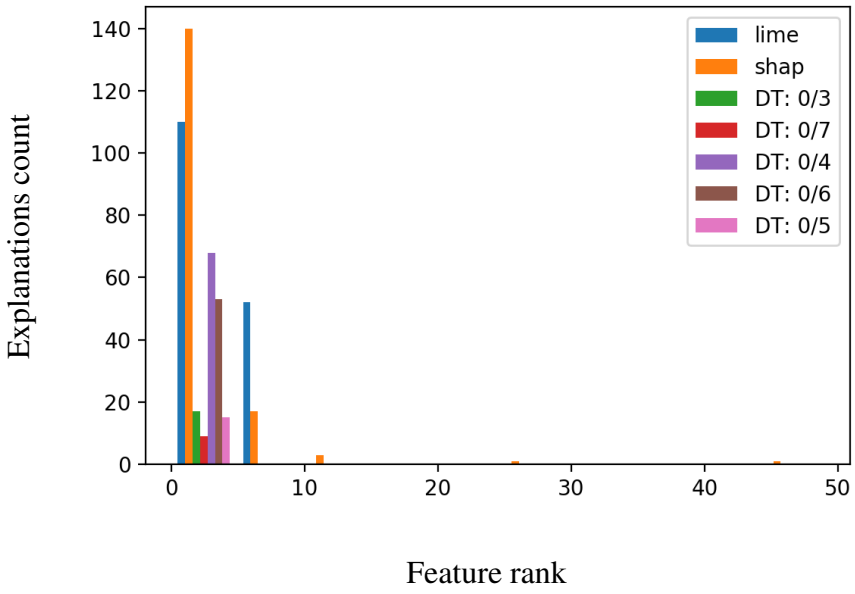


Figure 29. `HH_L3_weight` feature order-selection.

Another sample of these results in feature `H_L0.01_mean`, which is also often used for explanations. In Figure 30 presented that LIME mostly use `H_L0.01_mean` as top 1 feature for explaining some classes. It is specified in previous results in "LIME Feature selection" section. These are "ack" and "udp" classes. SHAP also has a high selection of this feature on higher ranks. In explanation path it is mostly on the third place.

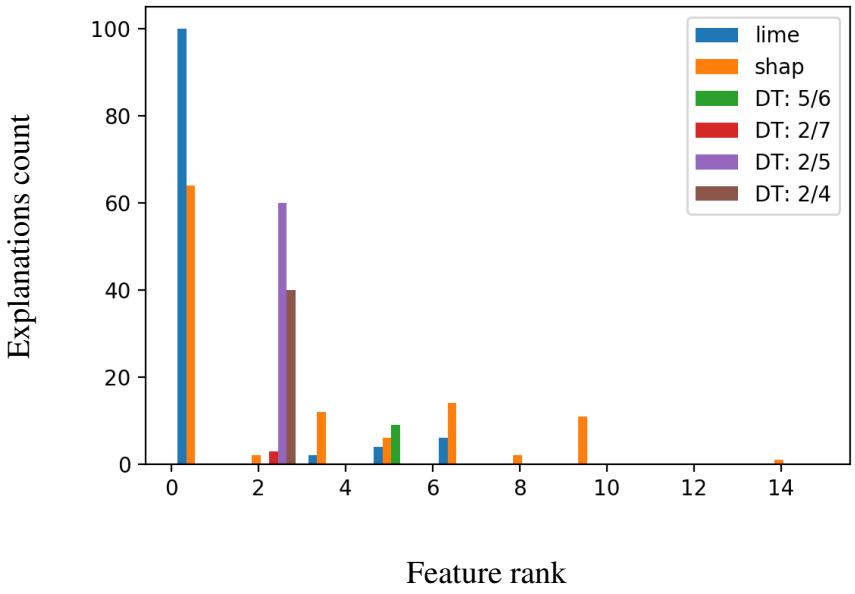


Figure 30. `H_L0.01_mean` feature order-selection .

Current experiment demonstrates the difference in feature ranks and selection frequency of these ranks. These results is a great overview of LIME, SHAP and a simple explanation path data. There is never a high variance in ranks between explanation methods for the highest bars. This indicates, that explanation path feature ranks is similar to the interpreters' feature ranks. A cyber security analyst can use such data to verify if current testing interpreter is producing similar feature ranks as some simple method like explanation path. It is much easier to verify the quality of Decision Tree structure, than compare interpreters rank results with more complex explainers.

4.5 Metrics

4.5.1 Consistency

The experiment with consistency metric used KNN and Decision Tree model. Both models have high prediction accuracy, above 98%. Thus, they considered to have similar prediction outputs. There were no specific parameters for Decision Tree, and the model has no limitations. KNN model was trained with the "number of neighbors" parameter, that was equal to five. A random set of samples is selected and explained by LIME and SHAP. Each explainer used two classification models and stored the results. The goal is

to compare the explanations of different models.

- LIME consistency results
  LIME often puts a lot weight on the feature in Decision Tree explanation path, which is not present in KNN explanation. Features that has relatively same order has similar weights. Feature weight direction is usually the same, but not in a 100% cases(see Figures 31-32). In the Figures 31 and 32 presented samples of two explanations with different models.

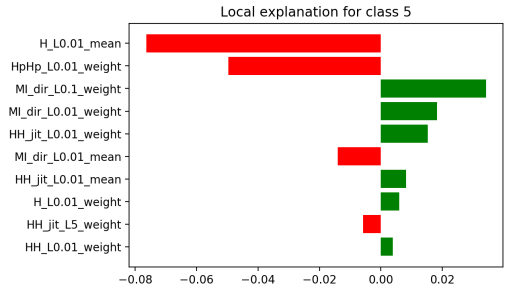


Figure 31. Decision Tree model.

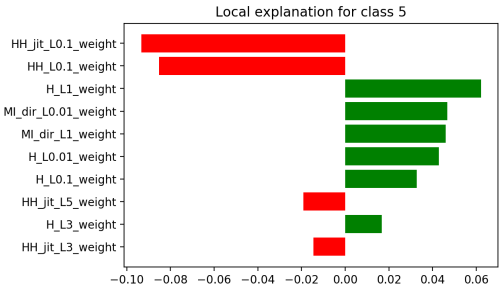


Figure 32. KNN model.

It is difficult to collect any specific data about these explanations because they are very inconsistent. Explanation are considered inconsistent because feature rank and feature weights are very different between various explanations. There is no two explanations from different models, that fully matching feature ranks and feature weights. For this experiment, some of the metrics were: percentage of average features matching in both models and percentage of average feature weight difference. Average values are calculated only for matching features in explanations.

For LIME this results are:
38.57% features matching in average
130.19% average feature weights difference

- SHAP consistency results
  For SHAP, the picture is very similar to LIME. Explanations are very different, and metrics numbers that are calculated for LIME looks even worse in this case. The number of features in explanations is often different. Sample results and numbers

below

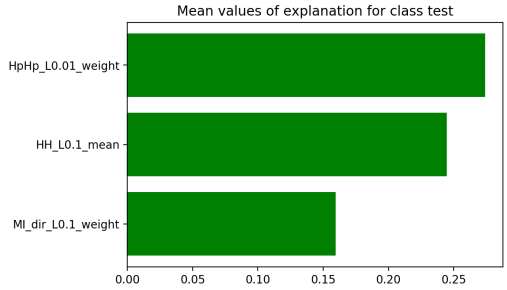


Figure 33. Decision Tree model.

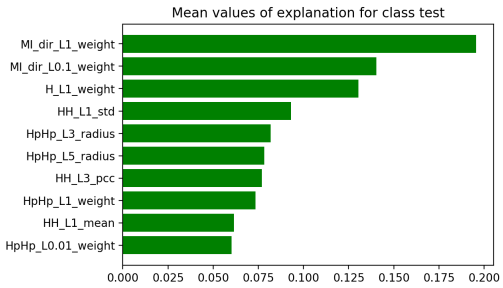


Figure 34. KNN model.

SHAP feature metrics:
17.15% features matching in average
168.28% average feature weights difference

LIME and SHAP are not consistent.

4.5.2 Novelty

The objective of novelty metric is to compare explanations of points, that are close and far away from the main distribution. Both LIME and SHAP showed impressive novelty results.
LIME explanations for "grouped" and "far away" points are different, but the first top features in the explanation usually match, and they have similar feature weight. They are the most significant features which define the explanation itself.
SHAP has similar results, where most significant features remain the same for different points explanation. However, for "far away" points, SHAP explanation usually contains some extra features which are absent in "grouped" explanation.

Since top features explanations are similar, and these features have the highest influence in explanation, LIME and SHAP could be considered to satisfy novelty metric.

4.5.3 Sensitivity

After conducting experiments upon several pairs of instances, which have remarkably similar features except for one which pushed one of the instances to another class - the results are all very similar. They show that both LIME and SHAP have high sensitivity. Script for picking these pairs found only three pairs in a given dataset of 80 000 samples. Results of one of the samples:

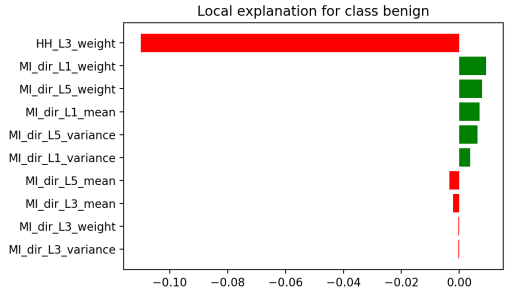


Figure 35. LIME explanation.

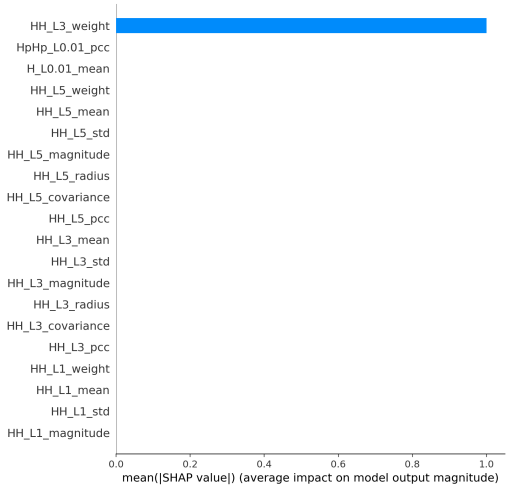


Figure 36. SHAP explanation.

This is an example of running a metric experiment on the samples which differ in one feature `HH_L3_weight`. One belongs to class benign, another to class "tcp". The other two pairs or samples involve features `HpHp_L0.01_weight` and `MI_dir_L0.01_mean`, which have similar results and are also included in explanation and have the highest ranks. Results for the samples, that differed in `HpHp_L0.01_weight` feature presented in the Figures 37 and 38.

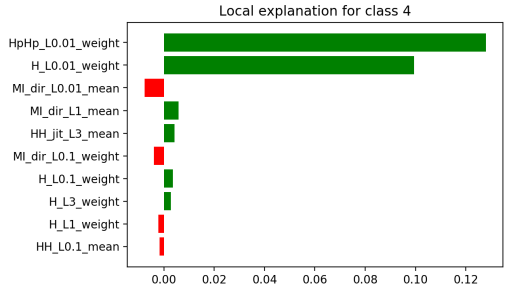


Figure 37. LIME explanation.

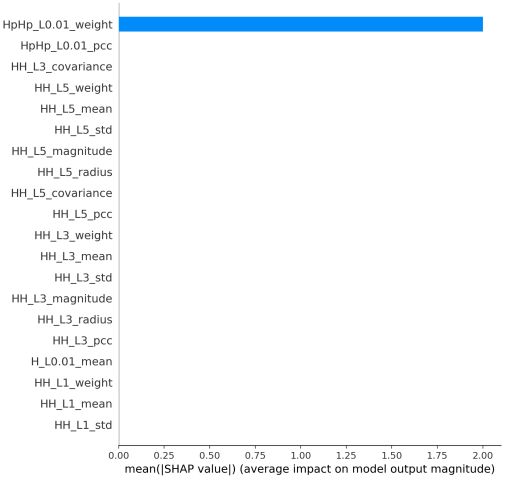


Figure 38. SHAP explanation.

LIME and SHAP are sensitive. The metric definition said that the feature should be present in explanation and has nonzero influence but for LIME and SHAP this feature is on the top of the explanation and has the strongest influence.

4.5.4 Stability

The results for this metric experiment are mostly taken from previous results. There is much data about points explanations and it was easy to gather some data about very similar samples. In the Figures 39-42 presented sample results of LIME and SHAP explanation for stability metric.

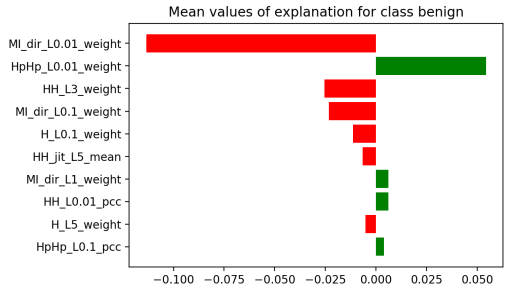


Figure 39. LIME benign explanation.

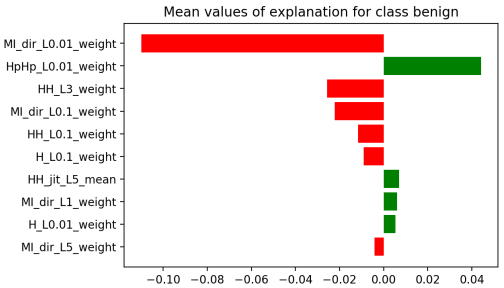


Figure 40. LIME benign explanation.

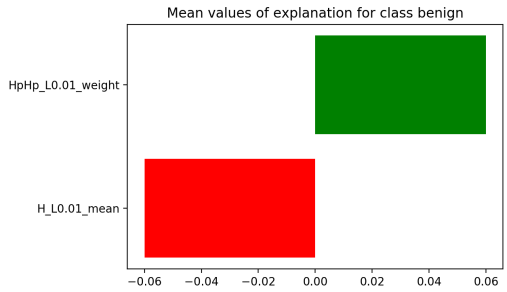


Figure 41. SHAP benign explanation.

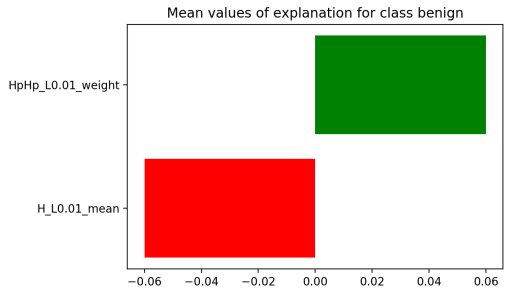


Figure 42. SHAP benign explanation.

In the Figures 39 and 40, 9/10 features are matching in both explanations. First top five features have the same rank and similar weights. The rest of features in explanation have mixed ranks, but the weights are similar as well.

LIME and SHAP are stable. In this case, SHAP shows better stability results, than LIME, and their weights are very similar. With LIME, the picture is slightly different, as most explanations are the same, except for a few. LIME explanations have a little variance in feature weights as well as some of them are entirely different, but these are rather exceptional cases. Nevertheless, there are very few of them, so LIME still can be considered stable. To evaluate all the cases for LIME and SHAP, a mean values and standard deviation of matching features is computed and presented in the Table 15. Selected are the top five features used by LIME and SHAP, plus the feature `H_L0.01_mean`, often used by SHAP, but not LIME.

Feature name	Method	Mean	Std
`MI_dir_L0.01_weight`	LIME	-0.127439	0.000539
`HpHp_L0.01_weight`	LIME	0.047984	0.000392
	SHAP	0.060236	0.000042
`HH_L3_weight`	LIME	-0.023932	0.000117
`MI_dir_L0.1_weight`	LIME	-0.024093	0.000335
`H_L0.1_weight`	LIME	0.0010993	0.000814
`H_L0.01_mean`	SHAP	-0.060052	0.000019

Table 15. Stability metric features data.

4.6 Levenshtein distance

A solution to calculating Levenshtein distance between explanations is to transform an explanation into a string. There was no possibility to include feature weights in this transformation, so the distance between explanations will involve only features themselves and their order. It is barely possible to fit 115 features into ASCII characters. Each feature is replaced with an ASCII character, which transformed an explanation for instances into a string. A sample of the string representation of the explanation for a random sample 48244:

48244: {'shap': '+-#*!"/,AY', 'lime': '($*!#"/3?:'}

Having two strings, it is possible to calculate Levenshtein distance. Since one can only calculate distance between 2 instances - the only option is to gather average results, which means that solution to the experiment is a set of mean values of Levenshtein distances for different sets of points. Also, standard deviation values are included for each mean value.

The results for this experiment are divided into two sections. First one is where Levenshtein distances is calculated within one class, and another one is with distances between different classes.

- Within a class
  Results are the mean values of distances between samples of one class for LIME and SHAP. All the samples and explainer configurations have been taken from the previous "one class" experiments.
  Distances for LIME are very high, considering that the maximum number possible is 10. For SHAP, they look a little lower, but standard deviation values are quite high compare to the distance values. Another experiment is required to see distances between classes, and then one can compare these results.
- Between classes
  To calculate Levenshtein distances between samples of different classes, ten samples from each class have been picked randomly, and the distance between them is calculated. Results are gathered in a way that for every class collected a set of mean values and standard deviations, just like from previous "within a class"

Class name	LIME mean	LIME std	SHAP mean	SHAP std
ack	9.0	0.632455	1.3	0.900001
benign	8.0	1.897366	0.0	0.0
combo	6.9	1.0440307	3.3	2.1931712
junk	6.7	1.345362	5.5	2.291288
syn	6.0	1.0	0.2	0.600001
scan	5.6	1.428286	0.4	0.800002
tcp	5.9	1.3	0.0	0.0
udp	8.0	0.4472136	3.7	2.1931712

Table 16. Levenshtein distance within a class.

results for each other class. Since there are eight classes and for each of them, there's a set of results for every other class - that makes 56 tables just like the one above. It is not efficient to present all of these here, just a summary of the results.

Some of the results look like this:

ack
lime mean distance to benign 9.7
lime std: 0.45825756949558394
shap mean distance to benign 2.6
shap std: 0.48989794855663565

scan
lime mean distance to ack 9.9
lime std: 0.3
shap mean distance to ack 2.6
shap std: 0.48989794855663565

Summary and conclusion - there is no connection between these distances; these are just some specific numbers for every pair of classes. It is important to check if any distance between the two classes is lower than the distance between samples of a single class. That would mean that explanations of two different classes vary less than explanations of a single class. However, no such cases are found. With these results, it is possible to gather information about which classes have the most similar explanations, if such data is required.

5 Conclusion

The main conclusion presented in a table, where all experiments and interpretation results are summarized.

	LIME	SHAP
Feature selection(all class) max percentage	79.75%	34.38%
Feature selection(one class) max percentage	100%	100%
Feature weights(all class)	high variance	high variance
Feature weights(one class)	low variance	high variance
Feature order	consistent	inconsistent
Explanation around decision boundary	consistent	inconsistent
Consistensy metric	No	No
Novelty metric	Yes	Yes
Sensitivity metric	Yes	Yes
Stability metric	Yes	Yes
Levenshtein distance within class values	High	Low
Levenshtein distance within class std	Low	High
Levenshtein distance between classes values	Generally high	Very different from class to class
Levenshtein distance between classes std	Generally low	Generally low

Table 17. Comparison table.

Interpretation for some of the results: Terms "low" and "high" used relatively. They represent the general behavior of specific values. "Low" means that values are close to a minimum possible value and "high" - close to the highest possible.

- "Feature selection(all class) max percentage" - maximum percentage of feature selection frequency in explanation by a specific algorithm. It shows that for "global" case, LIME and SHAP do not have a feature that is used for all explanations.

- "Feature selection(one class) max percentage" - show that every algorithm has a specific feature, used for explaining specific class.
- "Feature weights(all class)" - shows that features used by LIME and SHAP are very different for "all class" explanations.
- "Feature weights(one class)" - shows that features used by LIME for "one class" explanations have lower variance than SHAP feature weights.
- "Feature order" - among various explanations, LIME usually has the same features on the same orders in explanation, that is why it can be considered "consistent". SHAP assigns different order to features more often.
- Explanation around decision boundary - described in the "Explaining points close to decision boundary".
- All metrics - described in the "Metrics".
- Levenshtein distances - described in the "Levenshtein distance". When it is specified "generally high" or "generally low" - it means that results are too different from being classified as "high" or "low". In one case with SHAP, it is specified "Very different from class to class" because Levenshtein distances had most various values and can neither be classified as "high" nor "low" nor anything in between.

All the experiments and results provide a solution to the initial problems:

- Classifier selection
  For the LIME and SHAP it makes no difference which classification model to explain. All inputs are considered as a "black box" model. Decision Tree model is considered an explainable model, and it was selected because of two reasons. Firstly, it is a simple model from computer, which takes less time to be explained; and it is a simple model for human, as the Decision Tree structure is intuitive and every classification can be followed though the tree. Secondly, it produces explanation path, which is considered an explanation for a particular instance. Explanation path was used to compare feature ranks with LIME and SHAP.
- Feature selection
  Fisher score is calculated for every feature, and later used in feature analysis for LIME and SHAP. Interpretation methods have their own feature selection techniques and this study compared it with an external method, such as Fisher score.

- LIME/SHAP explanation analysis
  The major part of experiments were targeting LIME and SHAP explanations analysis. Feature selection behavior of both explainers is similar. None of them has a feature, which is used every time for "all class" explanations. And both of them have specific features for explaining every class in particular. Features selected to explain separate classes does not match between LIME and SHAP. Feature weights for LIME are more similar within "one class" explanations and less similar for "all class". For SHAP, they are dispersed for both "all class" and "one class" explanations. LIME produces similar explanations for points close and far from the decision boundary. SHAP explanations have only common feature rank, but feature weights and the number of selected features are different.
- Comparing LIME/SHAP explanations with Decision Tree
  Comparing feature ranks of LIME, SHAP and Decision Tree showed, that ranks are similar. Ranks represent feature importance, and even though explanation path of Decision Tree does not have feature weights - it can be used to verify the correct rank order in a comparable explainers.
- Metrics
  Both LIME and SHAP satisfy novelty, sensitivity and stability metrics. And do not satisfy consistency metric. Levenshtein distance analysis showed the differences(distances) of each LIME and SHAP explanations within a single class and between different classes. On average LIME explanations have high distances between different classes, and within a single class. While SHAP on average has low distance within a single class explanations and different distances between different classes. This means, that some classes have lower distances between their explanations and some have higher.

The next step to continue this study could be to experiment with a different dataset. On a different dataset LIME and SHAP could have different results of some metrics used in this thesis. Also, it is essential to analyze different types of classifiers. Classification models in this thesis worked with eight classes, but one needs to analyze if the explanation of a model with higher number of classes will be different.

References

[1] *Introduction to machine learning*, https://www.educba.com/introduction-to-machine-learning/, First paragraph.

[2] G. K. Zhicong Qiu David J. Miller, *Flow Based Botnet Detection through Semi-supervised Active Learning*. [Online]. Available: http://www.cse.psu.edu/research/publications/tech-reports/2016/CSE-16-010.pdf.

[3] T. L. Vaibhav Nivargi Mayukh Bhaowal, *Machine Learning Based Botnet Detection*. [Online]. Available: http://cs229.stanford.edu/proj2006/NivargiBhaowalLee-MachineLearningBasedBotnetDetection.pdf (visited on 2018-09-03).

[4] M. Alauthman, *An efficient reinforcement learning-based Botnet detection approach*. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S108480451930339X.

[5] A. Nayak, *Idea Behind LIME and SHAP*. [Online]. Available: https://towardsdatascience.com/idea-behind-lime-and-shap-b603d35d34eb.

[6] E. J. Dylan Slack Sophie Hilgard, *Fooling LIME and SHAP: Adversarial Attacks on Post hoc Explanation Methods*. [Online]. Available: https://arxiv.org/pdf/1911.02508.pdf.

[7] OnClick360, *Interpretable machine learning with Lime+ELI5+SHAP+InterpretML*. [Online]. Available: https://www.onclick360.com/interpretable-machine-learning-with-lime-eli5-shap-interpret-ml/.

[8] H. B. Alejandro Guerra-Manzanares Sven Nõmm, *Towards the Integration of a Post-hoc Interpretation Step into the Machine Learning Workflow for IoT Botnet Detection*. [Online]. Available: https://ieeexplore.ieee.org/document/8999281.

[9] Y. Meidan, *detection of IoT botnet attacks N BaIoT*. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/detection_of_IoT_botnet_attacks_N_BaIoT (visited on 2018-03-19).

[10] S. R. Riccardo Guidotti Anna Monreale, *A Survey Of Methods For Explaining Black Box Models*. [Online]. Available: https://arxiv.org/pdf/1802.01933.pdf.

[11] J. S. C. Diogo V. Carvalho Eduardo M. Pereira, *Machine Learning Interpretability: A Survey on Methods and Metrics*. [Online]. Available: https://www.mdpi.com/2079-9292/8/8/832/pdf.

[12] B. F. Adrien Bibal, *Interpretability of Machine Learning Modelsand Representations: an Introduction*. [Online]. Available: https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2016-141.pdf.

[13] A. B. B. Frénay, *Interpretability of Machine Learning Modelsand Representations: an Introduction*, page 81. [Online]. Available: https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2016-141.pdf.

[14] S. Lowry and G. Macpherson, *British medical journal (Clinical research ed.)* 296(6623):657. (visited on 1988).

[15] M. S. C. Carter E. Renuart, *The credit card market and regulation: In need of repair*, 10:23. (visited on 2006).

[16] S. R. Riccardo Guidotti Anna Monreale, *A Survey Of Methods For Explaining Black Box Models*, pages 3-5. [Online]. Available: https://arxiv.org/pdf/1802.01933.pdf.

[17] S. R. Riccardo Guidotti Anna Monreale, *A Survey Of Methods For Explaining Black Box Models*, pages 6. [Online]. Available: https://arxiv.org/pdf/1802.01933.pdf.

[18] B. C. M. F. Miles Q Li, *I-MAD: A Novel Interpretable Malware Detector Using Hierarchical Transformer*. [Online]. Available: https://arxiv.org/pdf/1909.06865.pdf.

[19] M. Q. L. Benjamin C M Fung, *I-MAD: A Novel Interpretable Malware Detector Using Hierarchical Transformer*, page 8. [Online]. Available: https://arxiv.org/pdf/1909.06865.pdf.

[20] S. S. Marco Tulio Ribeiro Carlos Guestrin, *Anchors: High-Precision Model-Agnostic Explanations*. [Online]. Available: https://homes.cs.washington.edu/~marcotcr/aaai18.pdf (visited on 2018).

[21] C. G. Marco Tulio Ribeiro Sameer Singh, *Model-Agnostic Interpretability of Machine Learningv*. [Online]. Available: https://arxiv.org/pdf/1606.05386.pdf.

[22] C. G. Sameer Singh Marco Tulio Ribeiro, *Anchors: High-Precision Model-Agnostic Explanations*. [Online]. Available: https://homes.cs.washington.edu/~marcotcr/aaai18.pdf (visited on 2018).

[23] S.-I. L. Scott M. Lundberg, *A Unified Approach to Interpreting Model Predictions*. [Online]. Available: https://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf (visited on 2017).

[24] S. I. L. Scott M Lundberg, *A Unified Approach to Interpreting Model Predictions*, pages 4-5. [Online]. Available: https://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf (visited on 2017).

[25] E. Štrumbelj and I. Kononenko, *Explaining prediction models and individual predictions with feature contributions*, pp. 647–665. (visited on 2014).

[26] S. S. Anupam Datta and Y. Zick, *Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems*, pp. 598–617. (visited on 2016).

[27] S. F. Bryce Goodman, *European Union regulations on algorithmic decision-makingand a "right to explanation"*. [Online]. Available: https://arxiv.org/pdf/1606.08813.pdf.

[28] W. N. van Wieringen, *Lecture notes on ridge regression*. [Online]. Available: https://arxiv.org/pdf/1509.09169.pdf.

[29] S. M. Lundberg, *From local explanations to global understanding with explainable AI for trees*. [Online]. Available: https://www.nature.com/articles/s42256-019-0138-9.epdf.

[30] N. H. David Dao Frances Ann Hubis, *Towards Efficient Data Valuation Based on the Shapley Value*. [Online]. Available: https://arxiv.org/pdf/1902.10275.pdf.

[31] Y. M. air Meidan Michael Bohadana, *N-BaIoT: Network-based Detectionof IoT Botnet AttacksUsing Deep Autoencoders*. [Online]. Available: https://arxiv.org/pdf/1805.03409.pdf.

[32] R. Gandhi, *Support Vector Machine — Introduction to Machine Learning Algorithms*. [Online]. Available: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47.

[33] D. B. Gongde Guo, *KNN Model-Based Approach in Classification*. [Online]. Available: https://hunch.net/~coms-4771/quinlan.pdf.

[34] J. QUINLAN, *Induction of Decision Trees*. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.2.815&rep=rep1&type=pdf.

[35] G. M. I. CHAO-YING JOANNE PENG KUK LIDA LEE, *An Introduction to Logistic Regression Analysis and Reporting*. [Online]. Available: https://www.researchgate.net/profile/Joanne_Peng2/publication/242579096_An_Introduction_to_Logistic_Regression_Analysis_and_Reporting/links/0deec5374c228b7fa1000000/An-Introduction-to-Logistic-Regression-Analysis-and-Reporting.pdf.

[36] K. B. A. Jović and N. Bogunović, *A review of feature selection methods with applications*. [Online]. Available: https://bib.irb.hr/datoteka/763354.MIPRO_2015_JovicBrkicBogunovic.pdf (visited on 2015).

[37] M. L. McHugh, *The Chi-square test of independence*. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3900058/.

[38] J. H. Quanquan Gu Zhenhui Li, *Generalized Fisher Score for Feature Selection*. [Online]. Available: https://arxiv.org/ftp/arxiv/papers/1202/1202.3725.pdf (visited on 2012).

[39] Z. L. Quanquan Gu Jiawei Han, *Generalized Fisher Score for Feature Selection*, page 2. [Online]. Available: https://arxiv.org/ftp/arxiv/papers/1202/1202.3725.pdf (visited on 2012).

[40] J. S. C. Diogo V. Carvalho Eduardo M. Pereira, *Machine Learning Interpretability: A Survey on Methods and Metrics*, page 17. [Online]. Available: https://www.mdpi.com/2079-9292/8/8/832/pdf.

[41] E. M. P. Diogo V. Carvalho Jaime S. Cardoso, *Machine Learning Interpretability: A Survey on Methods and Metrics*, page 18. [Online]. Available: https://www.mdpi.com/2079-9292/8/8/832/pdf.

[42] D. Mukhopadhyay, *Levenshtein Distance Technique in Dictionary Lookup Methods: An Improved Approach*. [Online]. Available: https://www.researchgate.net/publication/48180222_Levenshtein_Distance_Technique_in_Dictionary_Lookup_Methods_An_ImprovedApproach.

[43] X. Ying, *An Overview of Overfitting and its Solutions*. [Online]. Available: https://iopscience.iop.org/article/10.1088/1742-6596/1168/2/022022/pdf.