Trustworthy and Interpretable Machine Learning for Network Anomaly Detection

Yuangan Zou
University of Toronto
Toronto, ON, Canada
bill.zou@mail.utoronto.ca

ABSTRACT

Network anomaly detection plays a crucial role in ensuring the stability and security of modern computer networks. In this study, we explore how to integrate machine learning approaches for this specific task. Several machine learning models are trained with a simulated military network (LAN) anomaly dataset in order to test and compare their performances.[1] Also, it is essential to ensure that the machine learning models used in networking align with network policies and regulations, therefore the machine learning models implemented have to be trustworthy and interpretable. Therefore, we will explore potential ways for interpreting the decisions made by machine learning models, understanding how the decisions are made and evaluating the models based on their interpretability. For the result of this study, we will present a comparison across all machine learning models implemented with criteria including anomaly detection accuracy, algorithm runtime as well as model interpretability.

KEYWORDS

Network anomalies, Trustworthy Machine Learning, Machine Learning Interpretability, Feature Engineering, Anomaly Detection, Algorithm runtime, and Machine Learning Model Complexity, Binary Classification.

1 INTRODUCTION

Some traditional methods for network anomaly detection include flow-based and statistical-based detection. [2] However, with complex and evolving attack patterns, statistical and rule-based methods are facing the limitation of not being able to capture and detect new, unseen attacks. Therefore, machine learning techniques for network anomaly detection are brought to the stage since they are able to generalize better to new and unseen network anomalies. This project focuses on addressing this need by developing a novel approach that combines the principles of trustworthiness and interpretability in machine learning models for network anomaly detection. Trustworthiness is achieved by ensuring that the model is reliable and is able to detect network anomalies with excellent accuracy within a reasonable running time. Interpretability is achieved by designing models that provide understandable explanations for their decisions, making it easier for network administrators to trust and act upon the detected anomalies as well as to ensure that the model implemented aligns with network policies and regulations.

The proposed approach contains a range of machine learning classifiers in order to classify network anomalies from normal network traffic. All classifiers are trained on the same dataset of network traffic data, which includes both normal and anomalous traffic patterns in order to learn the underlying features and patterns of network anomalies. To explore the robustness of models, we plan to do feature engineering by first visualizing the importance of each feature and selecting the most important ones to train the models for a second time. We observe the

difference in classification accuracy as well as the total running time of the models in order to pick the best trade-off between number features and model performance. Finally, we study the interpretability of our models through the use of explainable AI techniques, including Yellowbrick, ELI5, SHAP, Mixtend, PDPBox and InterpretML[3]. These techniques provide global explanations for the model's predictions. The explanations are then visualized and presented to network administrators, enabling them to understand the reasons behind the model's detections and take appropriate actions to mitigate network anomalies and enforce network regulations.

The evaluation metrics of our study include classification accuracy for detecting network anomalies as well as the F1 score[4], running time for the model to generate predictions and the interpretability of the model. To evaluate the interpretability, we will first use explainable AI techniques to visualize how the model makes predictions, and then perform a subjective quantitative analysis by manually assigning a score for each machine learning algorithm. The range of the score is 0 (not interpretable at all) to 4 (great interpretability).

2 RESEARCH QUESTIONS

Here is a list of research questions that will be addressed in this study.

1. How do different ML algorithms perform when detecting network anomalies in terms of detection accuracy and running time efficiency?

2. Is there a trade-off between the number of features and accuracy as well as running time?

3. How and why did my machine learning model predict a network activity as an anomaly?

4. Which machine learning model offers the highest accuracy, best interpretability (in a subjective view) and shortest running time?

3 BACKGROUND LITERATURE

Lindemann[4] explores the use of an LSTM model for network anomaly detection, it also includes a graph-based and transfer learning-based approach to highlight the potential of upcoming anomaly detection techniques. This study serves as a valuable resource for researchers interested in utilizing LSTM networks for anomaly detection in networks. However, this study does not have a comparison of performances between different model types but focuses only on LSTM. Also, the interpretability analysis is not included.

Wang[5] compares the traditional anomaly detection approaches in networking with machine learning approaches to introduce the challenges of anomaly detection in traditional networks. The study also reviews the implementation of ML in anomaly detection as well as the procedure and comparison of using different ML algorithms. It discusses the advantages and limitations of each approach and provides insights into their applications in different network environments. The design of this study is similar to our's but it does not address the concern of how interpretable the ML models are. For the evaluation of the ML models, this study focuses solely on the detection accuracy, however, the running time of ML algorithms also affects the performance of the network system. Our study is able to fill these gaps by including running time in our evaluation metrics and performing an interpretability analysis.

Dutta[6] presents an ensemble method that leverages deep models such as the Deep Neural Network (DNN) and Long Short-Term Memory (LSTM) and a meta-classifier (i.e., logistic regression) for the task of anomaly detection in networking. It explores a two-stage method in which data are preprocessed in the first step and then classified in the second step. The efficiency and accuracy of the approach are tested on heterogeneous datasets. This study evaluates different ML models on both their classification accuracy and running time for network anomaly detection tasks. The design of this study is similar to our first research question. However, this study lacks an analysis of the trade-off between the number of features and accuracy as well as running time. Also, the analysis of interpretability is not included.

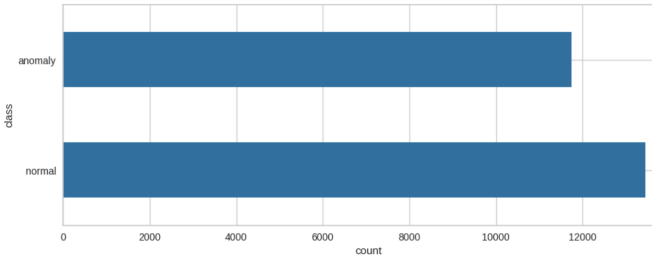
By looking at some previous work, we conclude that although there are works around evaluating the performance of different ML algorithms in both accuracy and running time for network anomaly detection tasks, there are few studies focusing on analyzing the interpretability of the ML algorithms as well as the impact of feature number on the model's accuracy and running time. Therefore, our study aims to fill this research gap by studying the trade-off between the number of features and accuracy as well as running time and performing an interpretability analysis using explainable AI techniques and subjective quantitative analysis.

4 METHODOLOGY

This study is implemented in Google Colab, which is a cloud-based coding environment provided by Google that allows users to write and run Python code.

4.1 Data Collection

We obtain our dataset from Kaggle[6]. It is an intrusion detection dataset which contains both normal and anomalous network traffic data. This dataset is relatively balanced, meaning that the numbers of anomaly and normal traffic are similar.



The dataset is created by dumping ICP/IP data for a network by simulating a typical US Air Force LAN. For each TCP/IP connection, 41 quantitative and qualitative features are obtained from normal and attack data (3 qualitative and 38 quantitative features). Each data sample has 41 features in total, such as protocal_type, dst_bytes, src_bytes and dst_addr. An example is as follows:

duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_ho
0	tcp	ftp_data	SF	491	0	0	0	0	0	...	

This dataset contains multiple network anomaly attacks such as DDoS, DoS and TCP SYN flood. However, in this dataset, the type of attacks is ignored and are all labelled as "anomaly", making this dataset suitable for doing binary classification.

4.2 Data Preprocessing

After data collection, we first separate the target column from the feature columns. We then remove some redundant features as well as Null values in the dataset. The target column contains two labels, normal and anomaly. We quantize them by assigning 1 to normal and 0 to anomaly. For the feature columns, we observe that although most features take numerical values, there are few features that are represented by non-numerical values (strings), such as the feature "protocol". In order to address this issue, we first extract the feature columns that take non-numerical values and encode them using the LabelEncoder class from

sklearn.preprocessing. After that, we perform standardization of all features, arriving at a zero mean and unit variance. Lastly, we split the dataset into training (80%) and testing sets (20%).

4.3 **Build Machine Learning Models**

We implement the coding for K-nearest-neighbour, logistic regression, decision tree and support vector machine classifiers, and then train them with our training dataset with full features and evaluate their classification accuracy using our testing set. The accuracy and running time for each model to generate the prediction for the testing set are recorded in order to compare their performance.

4.4 **Feature Engineering**

Then, we proceed to feature engineering by first visualizing the importance of each feature using a random forest, and then select the most important 3,5,10 and 20 features respectively and re-train the models in section 4.3 instead of using the full features option. We run the model evaluation again in order to examine whether there is a trade-off between the number of features and accuracy as well as running time for all models.

4.5 **Interpretability Study**

To study the interpretability of the models, we plan to apply slightly different techniques for each model since different models are interpreted differently. More specifically, we interpret our models using the following methods:

Model	Method
KNN	plot decision boundary
Logistic Regression	show feature importances, show threshold plot
Decision Tree	visualize tree, show feature importances
SVC	plot decision boundary

Finally, we will perform a subjective quantitative analysis by manually assigning an interpretability score for each model. The scores will be assigned by the author and his academic co-worker Qian Tang, who also specializes in computer engineering at the University of Toronto. The scoring criteria are as follows:

1. Model interpretability is easy to read and understand. (2 scores)
2. Feature importances can be shown. (2 scores)

We believe that showing feature importance is critical since it provides good insights into what dominates the detection of network anomaly detection. Also, it provides insights into potential ways of improving the ML model by selecting only the most dominant features in order to speed up the algorithm.

An inter-rater agreement analysis using Cohen's Kappa will be conducted to check whether the author and his co-worker have arrived at a general agreement. To end this study, we do a comparison across all models based on their accuracy, running time and interpretability.

5 **RESULTS**

In this section, we present the results of this study by answering our research questions.

RQ1: How do different ML algorithms perform when detecting network anomalies in terms of detection accuracy and running time efficiency?

We measure the performance of ML models by classification accuracy and running time. For this RQ, we train the models with the training set using full features. The classification accuracy is measured using the testing set and the running time is measured by timing the process in which

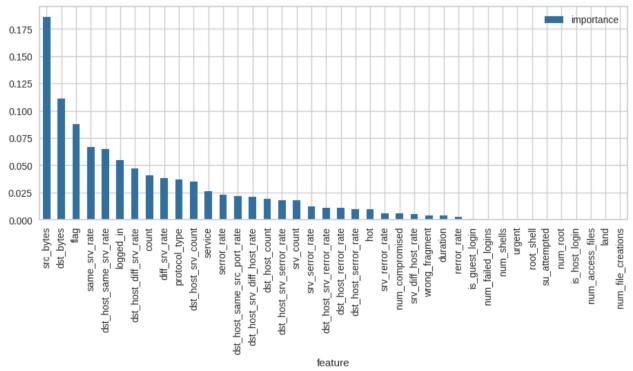
the model generates predictions for the testing set. The result is shown below:

Model	Prediction Accuracy	Running time (s)
K-Nearest-Neighbour	0.991665	3.461737
Logistic Regression	0.955944	0.957338
Decision Tree	0.995634	0.293381
Support Vector Machine	0.965866	5.714245

We can see that Decision Tree arrives at the highest classification accuracy 99.56% and it also has the quickest running time at 0.29 seconds. Therefore, we conclude that if we train the models with full features, the Decision Tree model has the best performance in both accuracy and running time.

RQ2: Is there a trade-off between the number of features and accuracy as well as running time?

In order to study the trade-off between the number of features and accuracy as well as running time, we first visualize the feature importance by using a random forest:



Then, we select the most important 3,5,10 and 20 features and retrain the models with the selected features. We perform the evaluation for the retrained models and obtain the following results:

Use 3 features:

Model performace with 3 feature:

	Model	Prediction Accuracy	Running time (s)
0	K-Nearest-Neighbour	0.966263	0.812013
1	Logistic Regression	0.877952	0.282223
2	Decision Tree	0.968248	0.115886
3	Support Vector Machine	0.878547	6.104423

Use 5 features:

Model performace with 5 feature:

	Model	Prediction Accuracy	Running time (s)
0	K-Nearest-Neighbour	0.965866	0.838276
1	Logistic Regression	0.869022	0.348504
2	Decision Tree	0.993848	0.124591
3	Support Vector Machine	0.904346	4.521597

Use 10 features:

Model performace with 10 feature:

	Model	Prediction Accuracy	Running time (s)
0	K-Nearest-Neighbour	0.985315	1.385398
1	Logistic Regression	0.950387	0.472660
2	Decision Tree	0.994443	0.161950
3	Support Vector Machine	0.955745	4.779296

Use 20 features:

Model performace with 20 feature:

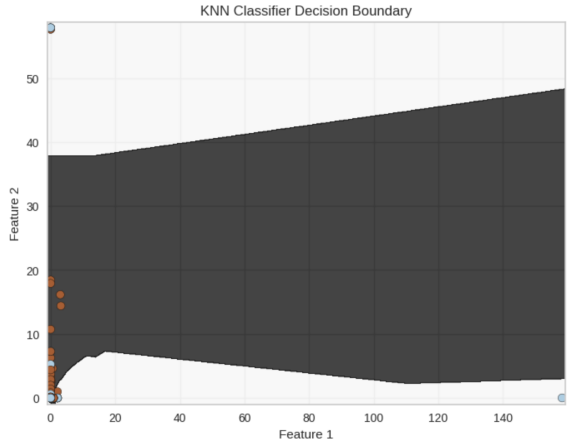
	Model	Prediction Accuracy	Running time (s)
0	K-Nearest-Neighbour	0.991070	2.423223
1	Logistic Regression	0.944037	0.861151
2	Decision Tree	0.995436	0.253118
3	Support Vector Machine	0.956142	7.441377

We can see that as we increase the number of features, the running time of all models increases as well. Also, the classification accuracy also increases with the number of features. If we compare the performance among the models, we see that Decision Tree always has the best accuracy as well as running time. Therefore, we conclude that there is a trade-off between the number of features and accuracy as well as running time. The more features we use, the higher the accuracy and the longer the running time. If we value the prediction accuracy and running time equally (each with 50% weight), for k-Nearest-Neighbour model, the best number of feature is 3, for Logistic Regression model, the best number of feature is 10, for Decision Tree, the best number of feature is 5 and for Support Vector Machine, the best number of feature is 10.

RQ3: How and why did my machine learning model predict a network activity as an anomaly?

a.KNN

We apply different methods to study the interpretability of our models. For k-Nearest-Neighbour, we plot the decision boundary. It is not possible to show the feature importance of the KNN model since it is a non-parametric machine learning algorithm that makes predictions based on the distances between data points in the feature space[7]. In order to visualize the decision boundary, we only use the two most important features so that we can plot it in a 2D graph.

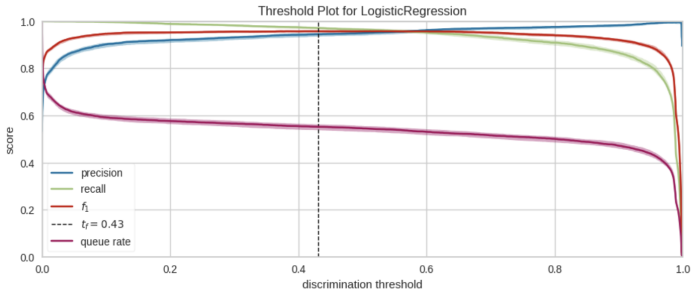


From the graph, we see that the data points at the bottom and top are classified as anomalies and the data points in the middle are classified as normal traffic. Therefore, we can conclude KNN detects network anomalies by measuring the distances between the examined data point and its surrounding data points, if there are more anomalies than normal data points in its surrounding, then it will be classified as anomaly. The interpretability of KNN model is not good since it would be hard to visualize the decision boundary if we have more than two features. Also, since the feature importance is not available, it is not intuitive which feature dominates the prediction.

b. Logistic Regression

For the Logistic Regression model, we interpret it by showing the feature importance as well as the threshold plot.

Weight[?]	Feature
+1.664	protocol_type
+1.626	dst_host_srv_count
+1.268	same_srv_rate
+0.730	is_guest_login
+0.411	diff_srv_rate
+0.377	srv_count
	… 10 more positive …
	… 10 more negative …
-0.358	flag
-0.396	num_compromised
-0.407	src_bytes
-0.475	serror_rate
-0.527	dst_host_serror_rate
-0.873	hot
-0.886	dst_host_same_src_port_rate
-0.899	dst_host_srv_serror_rate
-0.962	dst_host_count
-1.129	dst_host_same_srv_rate
-1.243	srv_serror_rate
-1.262	srv_rerror_rate
-1.268	wrong_fragment
-1.424	count

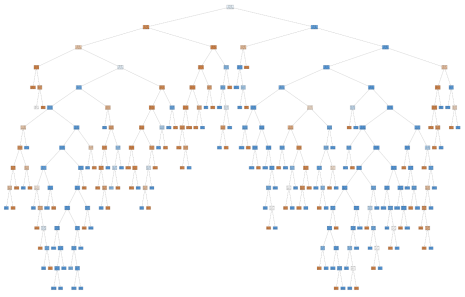


From the above results, we can see that the features “protocal_type”, “dst_host_srv_count” as well as “same_srv_rate” are dominating the detection of network anomalies. The best threshold for logistic regression arrives at 0.43, meaning that if a network traffic is predicted to have less than 43% probability of being normal, then it is an anomaly. The interpretability of Logistic Regression is better than KNN since we can visualize the importance of each feature in the anomaly detection process. Also, the threshold can show the decision-making process of the model, as it indicates the level of confidence required to make a prediction. This information can help understand the model's behaviour and how it makes predictions, which can be useful in explaining the model to stakeholders.

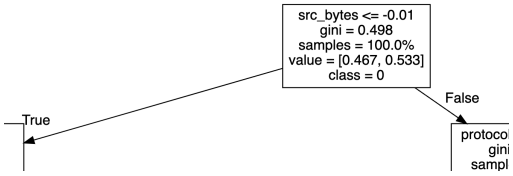
c. Decision Tree

For the Decision Tree model, we interpret it by showing the feature importance and visualizing the decision tree.

Weight	Feature
0.7817	src_bytes
0.0786	protocol_type
0.0546	dst_host_diff_srv_rate
0.0419	count
0.0210	dst_host_same_srv_rate
0.0153	dst_bytes
0.0052	flag
0.0013	diff_srv_rate
0.0003	same_srv_rate
0.0000	logged_in



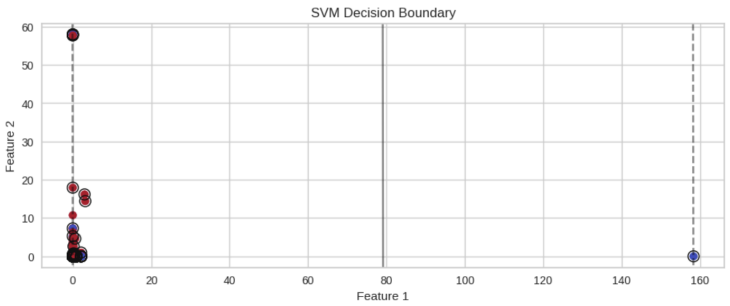
The decision tree shown here is large and we can zoom in to any of its nodes to obtain a clearer view, for example at the root:



From the above results, we can see that the features “src_bytes”, “protocol_type” as well as “dst_host_diff_srv_rate” are dominating the detection of network anomalies. By visualizing the decision tree, we can see that the prediction is made by splitting the nodes based on certain feature values. By displaying the tree structure and the decision-making process in a graphical format, it becomes easier to follow the logic and the sequence of the splits that led to a particular decision. The interpretability of decision tree is relatively good compared with logistic regression and KNN since it not only can show the feature importance but also is able to show a tree-structure-splitting plot so that it can be easily understood by non-technical stakeholders.

d. Support Vector Machine

The support vector machine model can be interpreted by plotting the decision boundary, similar to KNN. It is hard to show the feature importance for SVM since SVMs do not inherently provide feature importance scores like other algorithms such as decision trees. SVMs work by finding the optimal hyperplane that separates the data into different classes and the location of the hyperplane in the feature space is determined by the support vectors, which are a subset of the training data points that lie closest to the hyperplane[8].



From the above results, we can see that the decision boundary lies around in the middle, separating the two classes, the support vectors are plotted with a black circle. The interpretability of SVM is relatively bad since it cannot show the feature importance which offers important insights about what dominates the prediction. Also, similar to KNN, it would be hard to visualize the decision boundary if we have more than two features.

e. Interpretability scores for all models

The assigned interpretability scores for our models are shown below:

Model	Rater 1	Rater 2	Avg Score
KNN	2	2	2
Logistic Regression	3	4	3.5
Decision	4	4	4

Tree			
SVM	2	1	1.5

We can see that Decision Tree has the highest interpretability score and SVM has the lowest interpretability score. The Cohen's Kappa is calculated to be 0.33, indicating that a fair agreement has been achieved based on the following interpretation[9]:

Value of κ	Strength of Agreement
< 0.20	Poor
$0.21 - 0.40$	Fair
$0.41 - 0.60$	Moderate
$0.61 - 0.80$	Good
> 0.80	Very Good

RQ4: Which machine learning model offers the highest accuracy, best interpretability (in a subjective view) and shortest running time? We can see that from the answers to RQ1, 2 and 3, the Decision Tree model offers the best accuracy as well as the shortest running time. The interpretability of the Decision Tree model is also the best. Thus, we conclude that the best model in our study is the Decision Tree.

6 CONCLUSION

We present a study of using Machine Learning techniques to detect network anomalies, more specifically, we focus on network attacks that aim to utilize the vulnerabilities of our network system. With complex and evolving attack patterns, statistical and rule-based methods are facing the limitation of not being able to capture and detect new, unseen attacks. Therefore, machine learning techniques for network anomaly detection are brought to the stage since they are able to generalize better to new and unseen network anomalies.

Also, since it is critical to make sure the ML algorithms align with network policies and regulations, we need to understand the decisions made by ML models as well as why they make the decisions. Therefore, we are motivated to introduce an interpretability study with a focus on exploring how ML models make their predictions and how easy it is for humans to understand them.

We select four types of ML models for our study: KNN, Logistic Regression, Decision Tree as well as SVM. Our results indicate that the Decision Tree model performs the best in terms of both prediction accuracy and running time. Also, we observe that for all models, there is a trade-off between the number of features and accuracy as well as running time. The more features we use to train the model, the longer the running time and the higher the accuracy. Therefore, it worths careful consideration in choosing the number of features when training a ML algorithm in order to obtain the best performance. For the interpretability of models, we observe that the Decision Tree also has the best interpretability since we can not only visualize the feature importance but also can show the tree-structure-splitting of the decision-making process. Visualizing feature importance offers a good view of what dominates the prediction of network anomalies, while the tree-structure-splitting makes it easy to follow the logic and the sequence of the splits that led to a particular decision.

The main contribution of our study is that we evaluate ML models for network anomaly detection tasks not only by prediction accuracy, but also we take running time as well as interpretability into consideration. We understand that it is essential to detect and respond to anomalies in real-time to prevent potential security breaches. If the ML model takes too long to process incoming network traffic data, it can lead to delays in anomaly detection and response. Also, since network anomaly detection is often used in sensitive environments such as finance or healthcare, regulatory compliance is critical. Thus, we believe that the ML models

must be explainable, and their decisions must be easily understood to meet regulatory requirements.

7 FUTURE WORK

There are some limitations in our study which can be addressed by future work.

a. External Validity

One limitation is that we didn't consider the impact of network environment. The dataset used in this study is created by dumping ICP/IP data for a network by simulating a typical US Air Force LAN. If we use a dataset created in a mobile network, it is possible that our study cannot generalize well. Therefore, a future study can try to reproduce this study by using a dataset generated in a different network environment in order to verify whether the network environment has any impact on the final results. Also, the ML models selected for evaluation may not be representative of the broader range of ML models available, potentially leading to results that do not generalize to other ML models.

b.Internal Validity

A threat to internal validity is about interpretability. The interpretability of models is evaluated using different methods, therefore, it is possible that we did not choose the best method to interpret each model we implemented in this study. A future study could reproduce this process by using the same method to evaluate the interpretability of all models so that when we compare the models in terms of interpretability, the result is more reliable. Also, since we defined our own scoring system to rate the interpretability scores for each model, the scoring criteria might be biased. A better approach is to conduct a survey with more participants in order to construct a more comprehensive scoring system.

REFERENCES

[1] Galaxyh. (2018). KDD Cup 1999 Data [Data set]. Kaggle. https://www.kaggle.com/datasets/galaxyh/kdd-cup-1999-data

[2] A comparative analysis of traditional and deep learning-based anomaly ... (n.d.). Retrieved April 12, 2023,from https://www-live.dfki.de/fileadmin/user_upload/import/10754_comparative_anomaly_detection_ICMLA.pdf

[3] Jain, A. (2020, March 23). 6 Python Libraries to Interpret Your Machine Learning Model. Analytics Vidhya. Retrieved from https://www.analyticsvidhya.com/blog/2020/03/6-python-libraries-interpret-machine-learning-models/

[4] Lindemann, B., Maschler, B., Sahlab, N., & Weyrich, M. (2021). A survey on anomaly detection for technical systems using LSTM networks. Computers in Industry, 131, 103498. https://doi.org/10.1016/j.compind.2021.103498

[5] S. Wang, J. F. Balarezo, S. Kandeepan, A. Al-Hourani, K. G. Chavez and B. Rubinstein, "Machine Learning in Network Anomaly Detection: A Survey," in IEEE Access, vol. 9, pp. 152379-152396, 2021, doi: 10.1109/ACCESS.2021.3126834.

[6] Galaxyh. (2018). KDD Cup 1999 Data [Data set]. Kaggle. https://www.kaggle.com/datasets/galaxyh/kdd-cup-1999-data

[7] S. Raschka and V. Mirjalili, "Python Machine Learning," Birmingham, UK: Packt Publishing, 2017.

[8] "Support vector machine," Wikipedia, 2023. [Online]. Available: https://en.wikipedia.org/wiki/Support_vector_machine. [Accessed: Apr. 19, 2023].

[9] M. Samiullah, "Interpretation of Kappa Values," Towards Data Science, Aug. 08, 2019. [Online]. Available: https://towardsdatascience.com/interpretation-of-kappa-values-2acd1ca7b18f. [Accessed: Apr. 19, 2023].