A Dual Approach for Preventing Blackhole Attacks in Vehicular Ad Hoc Networks Using Statistical Techniques and Supervised Machine Learning

Abiral Acharya
Department of Electrical Engineering & Computer Science
University of Toledo
Toledo, Ohio, USA
abiral.acharya@rockets.utoledo.edu

Jared Oluoch
Department of Engineering Technology
University of Toledo
Toledo, Ohio, USA
jared.oluoch@utoledo.edu

2021 IEEE International Conference on Electro Information Technology (EIT) | 978-1-6654-1846-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/EIT51626.2021.9491885

***Abstract*—Vehicular Ad Hoc Networks (VANETs) have the potential to improve road safety and reduce traffic congestion by enhancing sharing of messages about road conditions. Communication in VANETs depends upon a Public Key Infrastructure (PKI) that checks for message confidentiality, integrity, and authentication. One challenge that the PKI infrastructure does not eliminate is the possibility of malicious vehicles mounting a Distributed Denial of Service (DDoS) attack. We present a scheme that combines statistical modeling and machine learning techniques to detect and prevent blackhole attacks in a VANET environment.**

Simulation results demonstrate that on average, our model produces an Area Under The Curve (ROC) and Receiver Operating Characteristics (AUC) score of 96.78% which is much higher than a no skill ROC AUC score and only 3.22% away from an ideal ROC AUC score. Considering all the performance metrics, we show that the Support Vector Machine (SVM) and Gradient Boosting classifier are more accurate and perform consistently better under various circumstances. Both have an accuracy of over 98%, F1-scores of over 95%, and ROC AUC scores of over 97%. Our scheme is robust and accurate as evidenced by its ability to identify and prevent blackhole attacks. Moreover, the scheme is scalable in that addition of vehicles to the network does not compromise its accuracy and robustness.

***Index Terms*—blackhole, machine learning, roc, auc, accuracy, f1-score AODV, VANETs**

I. Introduction

Vehicular Ad Hoc Networks (VANETs) is a form of Intelligent Transportation Systems (ITS) for connected and autonomous vehicles (CAV) where vehicles send safety and non-safety messages [1] to each other about road conditions. Communication in VANETs occurs in four ways: Vehicle to Infrastructure (V2I), Vehicle to Vehicle (V2V), Vehicle to Cloud (V2C), and Vehicle to Pedestrian (V2P). Collectively, this is known as Vehicle to Everything (V2X). In the United States, communication in VANETs is facilitated by a Security Credential Management System (SCMS) which relies on Public Key Infrastructure (PKI) [2]. SCMS is based on the Dedicated Short Range Communication (DSRC) standard and IEEE 802.11p Wireless Access in Vehicular Environments (WAVE) protocol [3]. Both DSRC and WAVE check for message integrity and authenticity but do not sufficiently identify and detect any misbehavior by attackers in the network.

Due to the ephemeral nature of VANETs, vehicles exchanging messages in the network are vulnerable to a Distributed Denial of Service (DDoS) attack. A blackhole attack is an example of a DDoS attack. It occurs when a vehicle falsely broadcasts it has the shortest path to the intended destination of a message. The network directs all messages to the compromised vehicle and the vehicle drops or forwards the messages at will. The consequences can be dire: road traffic accidents that can lead to death; road congestion; and unreliable messages. Fig.1 represents a black hole attack in a VANET environment. The blackhole attack interferes with the Route Request Packet (RREQ) and Route Reply (RREP) packet.

Machine learning techniques have been presented to aid in intrusion detection in vehicular environments. The authors in [1] apply Support-Vector Machine (SVM) and k-nearest neighbors (k-NN) algorithms to train their network to learn of wormhole attacks. Their results demonstrate that machine learning techniques can help prevent various attacks in a connected vehicle environment. In [4], the authors present a machine learning-based intrusion detection method that adapts to environmental changes especially when malicious vehicles overtake sensor units. The authors in [5] implement the SVM algorithm to show that machine learning mechanisms can improve reliability and communication efficiency in vehicular environments. The work by [6] shows that k-NN and SVM

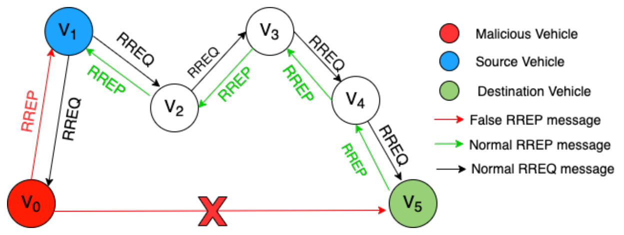


Fig. 1. Black hole attack

algorithms improve the overall detection rates of malicious vehicles by over 20%. In [7], results demonstrate that the Random Forest and J-48 classifiers perform comparatively better than other classifiers in isolating malicious nodes. In [8], the authors suggest that machine learning techniques can use the receiver power coherency metric to detect misbehaving nodes.

While existing literature presents various machine learning solutions for detecting malicious vehicles in VANETs, as far as we can tell, no current approach does a comparative study of the accuracy of five or more statistical and machine learning techniques for misbehavior detection. Accordingly, we present a scheme that combines statistical modeling and machine learning techniques to detect and prevent blackhole attacks in a VANET environment. We use SVM, Gradient Boosting, Gaussian Naive Bayes, k-NN, and Logistic Regression to compare the performance of each classifier against a data set generated using NS-3 simulator [9] [10] [11].

We collect enough data based on normal and malicious traffic flow to train and test different machine learning and statistical models and compare the performance of each model. We implement our model using Ad hoc On-Demand Vector Routing (AODV) protocol to: 1) classify incoming traffic flow, 2) identify and detect malicious vehicles, and 3) prevent a blackhole attack.

The performance metrics we use are accuracy, F-1 score, and Receiver Operation Characteristics (ROC) Area Under Curve (AUC). The data set consists of traffic flows under different scenarios where normal and malicious nodes are present. We use several parameters in training our dataset: number of normal vehicles, malicious vehicles, data transmission rate, packet size, number of packets, and the initial position of vehicles. We perform our simulation in NS3 under the IEEE 802.11p WAVE protocol with the two-rays ground propagation loss model. We extract 19 features and use analysis of variance (ANOVA) by computing the f-value as a feature selection technique to select the best 7 features.

Simulation results demonstrate that on average, our model produces an Area Under The Curve (ROC) and Receiver Operating Characteristics (AUC) score of 96.7% which is much higher than a no skill ROC AUC score and only 3.22% away from an ideal score. Considering all the performance metrics, we show that the SVM and Gradient Boosting classifier are more accurate and perform consistently better under various circumstances. Both have an accuracy of over 98%, F1-scores of over 95%, and ROC AUC scores of over 97%. Our model is scalable in that addition of more vehicles to the network does not negatively affect its accuracy.

The rest of this paper is organized as follows. Section II presents the machine learning and statistical models we implemented. Section III describes the design of our system. We present our results in Section IV. Finally, we conclude our work in Section V.

II. Machine Learning and Statistical Models

This paper implements the following machine learning and statistical models to detect malicious vehicles in our VANET environment.

A. Naive Bayes Classifier

Naive Bayes classifiers are based on Bayes theorem which "naively" assumes the conditional independence between every pair of feature values given the class variable values(y).

Bayes Theorem gives the following relationship between conditional probabilities of feature vector x_1 through x_n:

$$P(y|x_1, ..., x_n) = \frac{P(y)P(x_1, ..., x_n|y}{P(x_1, ..., x_n)} \tag{1}$$

The naive conditional independence assumption is:

$$P(x_i|y, x_1, ..., x_n) = P(x_i|y) \tag{2}$$

The classification rule for Naive Bayes classifier after all simplification is given as follow:

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i|y) \tag{3}$$

This classifier is good speed-wise but when it comes to estimation and prediction, it is less sophisticated than other classifiers. In this paper, we have used Gaussian Naive Bayes for classification that assumes the likelihood of features to be gaussian. [12].

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right) \tag{4}$$

B. Logistic Regression

Logistic Regression is a linear model which can be used for binary classification. When dealing with decent sample size, this classification model can outperform SVM and other ensemble learning models. It draws a response variable from the sample size which is the log of the odds of being classified in the i^{th} group of binary or multi-class response. A sigmoid function shown in (5) is used to model conditional probability distribution.

$$\sigma(Z) = \frac{1}{1 + \exp^{-z}} \tag{5}$$

The binary classification conditional probability function is as follow:

$$P(y_i = 1|x_i, w) = 1 - \frac{1}{1 + \exp^{(w^T x_i + b)}} \tag{6}$$

$$P(y_i = 0|x_i, w) = 1 - \frac{1}{1 + \exp^{(w^T x_i + b)}} \tag{7}$$

where w is a weight and b is a bias term for each input features [13].

C. K-nearest neighbors Classifier

The nearest neighbors model does not deal with gathering useful internal information from the dataset, but simply stores instances of training the dataset. For any unknown data point, it computes the distance between that point and known class points and classifies it as a class that has the most class representative in its neighborhood. In this paper, we implement the k-nearest neighbor algorithm where we select the value of k as 3. The reason for choosing an odd number is to avoid a tie situation while counting the votes [12].

D. Support Vector Machine

For binary classification, SVM constructs a hyperplane in a high dimensional space that separates the two classes. It tries to maximize the distance between the nearest training data points (support vectors) of both classes and the hyperplane (often called *margin*). The main goal of different SVM methods is to maximize the margin to reduce the generalization error of the classifier.

In this paper, we implemented Support Vector Classification (SVC) with *radial basis function* (rbf) as the kernel function which allows us to set the *gamma* value that helps to define how much influence a single training example has. SVC tries to solve the optimization problem by maximizing the margin and the output of decision function for a given sample x is:

$$\sum_{i\epsilon SV} y_i\alpha_i K(x_i, x) + b \tag{8}$$

where α_i is the dual coefficient, upper-bounded by a penalty term C, and $K(x_i, x)$ is the kernel function [12].

E. Gradient Boosting Classifier

Gradient boosting classifiers are a group of machine learning algorithms that combine many weak learning models to create a strong predictive model. Decision trees are generally used when doing gradient boosting. Gradient boosting builds trees one at a time, where each new tree helps to correct errors made by a previously trained tree. In this paper, we implement Sklearn's Gradient Boosting Classifier with a learning rate of 0.2, max depth of 1, and random state of 0 to achieve consistent results. All other parameters used default value. This type of ensemble learning generally performs better than other bagging classifiers when we are dealing with a decent amount of data and where computational complexity is not an issue [10].

III. System Design

A. Simulation Parameters

We use the NS-3 simulator to simulate normal and malicious VANET scenarios. NS-3 is well-suited for the VANET environment because all the modern standards for VANET communication are integrated into it. Also, it has topology-based routing protocols like AODV, Optimized Link State Routing Protocol (OLSR), and Destination-Sequenced Distance-Vector Routing (DSDV) already defined in its code-base. Under various traffic conditions, AODV appears to be the best performing protocol when we look at the metrics like Packet Delivery Ratio, throughput, and end-to-end delay [14]. To simulate blackhole attacks, we modified the AODV protocol to introduce malicious vehicular nodes and updated the routing table accordingly. We ran the simulation several times under varying conditions. The varying simulation parameters were: number of normal vehicles, malicious vehicles, data transmission rate, packet size, number of packets, and the initial position of vehicles. Table I shows the simulation parameters.

B. Dataset Generation

We utilize the flow-monitor tool present in NS-3 to extract useful data from our simulation. Flow monitor records every detail of traffic flow which can be used to analyze the performance of the simulation [15]. The output of each simulation is an Extensible Markup Language (XML) file that contains information regarding each flow. We use the Python Element Tree module to extract features from XML file to Comma-Separated Values (CSV) file. Our dataset contains numerical input features and categorical target labels. This is a binary classification problem. Table II shows the dataset characteristics.

C. Feature Selection

In total, we extracted 19 features. We used Analysis of Variance (ANOVA) by computing F-value as a feature selection technique to score each feature concerning its impact on the target. Looking at the difference in scores, we selected the best 7 features. The scores of all the features are summarised in Fig. 2. The eliminated features had very little to no impact on the target.

The final features selected were: source IP address, source and destination port, timeFirstRxPacket, timeLastRxPacket, lost packets, and throughput.

D. Performance Metrics

Machine learning can solve various problems related to classification and regression. Many metrics help one evaluate a model. In this paper, we only focus on binary classification metrics. The most straightforward metric to evaluate the binary classifier is classification accuracy. Accuracy is a good metric

TABLE I
Simulation Parameters

Parameters	Specifications
Platform	MacOS Big Sur 11.2.1
Network Simulator	Network Simulator-3.31
Mobility Model	Constant Position Mobility Model
Total no. of vehicles	4 - 50
Total no. of malicious vehicles	1 - 7
Data rate	500 Kbps - 1000 Kbps
Wave packet size	1024 bytes - 2048 bytes
No. of packets	5 - 65
Propagation delay model	Constant Speed Propagation
Propagation loss model	Two ray ground
IEEE scenario	VANET (IEEE 802.11p)
Routing Protocol	AODV

TABLE II
DATASET CHARACTERISTICS

Total records	5006
Targets	0 - normal, 1 - malicious
Total features	18
Target 0	3946
Target 1	1060
Training set	3505
Test set	1501

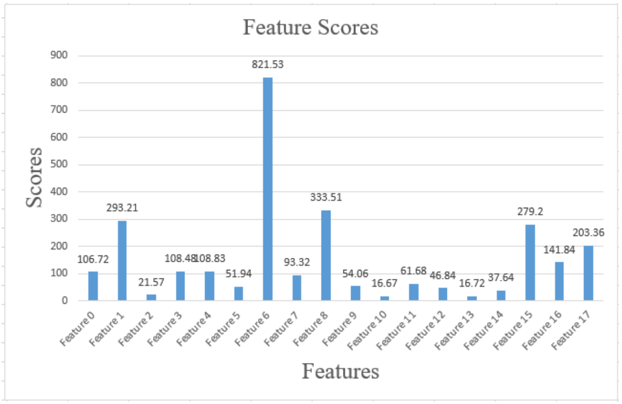
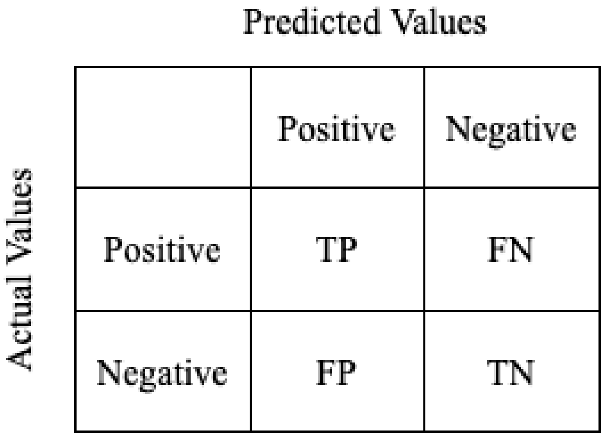


Fig. 2. Feature scores

but when the dataset consists of an imbalanced number of classes record, then only by predicting a high number of classes every time do we get a good accuracy score. That is why we shift our focus to other performance metrics like F1-score and ROC AUC score. All of the three metrics can be calculated when we form a confusion matrix. A confusion matrix is a contingency table that shows the difference between actual and predicted classes for a set of labeled examples [16]. Fig. 3 shows the representation of a confusion matrix for a binary classification problem.

- **TP:** It stands for True Positive. It is a correctly predicted positive or true value.
- **FP:** It stands for False Positive. It is an incorrectly predicted positive or true value.
- **FN:** It stands for False Negative. It is an incorrectly predicted negative or false value.
- **TN:** It stands for True Negative. It is an correctly predicted negative or false value.



Predicted Values

Actual Values		Positive	Negative
	Positive	TP	FN
	Negative	FP	TN

Fig. 3. Confusion Matrix

With the help of the confusion matrix, we can define other metrics for evaluating binary classifiers. Some of the metrics are as follows:

Accuracy: It is the ratio of correctly predicted values to the total values.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{9}$$

Recall: Recall gives us the score of correctly predicted positive values over all the positive values. It is also called Sensitivity or True Positive Rate (TPR).

$$Recall = \frac{TP}{TP + FN} \tag{10}$$

Precision: Precision is the ratio of correctly predicted positive values to the all positive predicted values.

$$Precision = \frac{TP}{TP + FP} \tag{11}$$

F1-score: It is the harmonic mean of Recall and Precision. The highest possible value for F1-score is 1.0. Higher F1-score signifies the better performance of a model.

$$F1 - score = \frac{2 * Recall * Precision}{Recall + Precision} \tag{12}$$

False Positive Rate: It is also called the false alarm rate. It is the ratio of incorrectly predicted positive values to the actual negative values. It is also called Inverted Specificity. It is also denoted as 1 - Specificity.

$$FPR = \frac{FP}{FP + TN} \tag{13}$$

Receiver Operating Characteristic (ROC) Curve: This is the plot of True positive percentage on the Y-axis against the False positive percentage on the X-axis. Once the classifier is run, a single point is plotted on the curve. Cross-validation is used to get a curve rather than a point [10]. In this paper, we use Area under the curve (AUC) to compare the performance of different classifiers. The highest possible value for the AUC score is 1.0. A higher AUC score signifies a better performance of a model.

IV. RESULTS

To develop the machine learning model for each classifier, we used Python's scikit-learn machine learning library [12]. In addition to this library, we also used NumPy and pandas library which made the handling of large datasets easier [17] [18]. The code was written on Jupyter Notebook [19]. The code was run individually for each classifier.

We split the dataset into 70% training and held out 30% as a testing set. The output or the classification metrics of all the classifiers is summarised in Fig. 4 and Fig. 5. The optimal number of features selected for better performance of all the

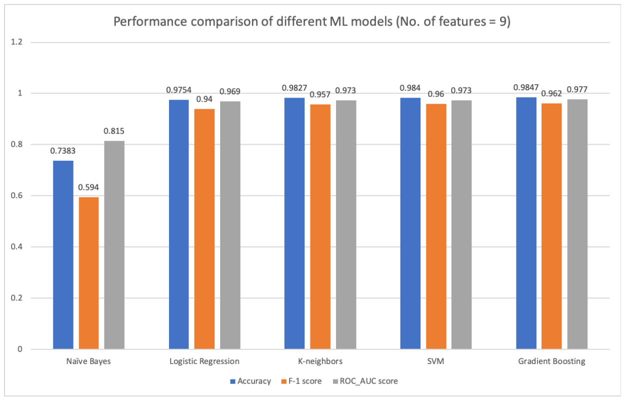


Fig. 4. Comparison of different classifiers (No. of features selected = 9)

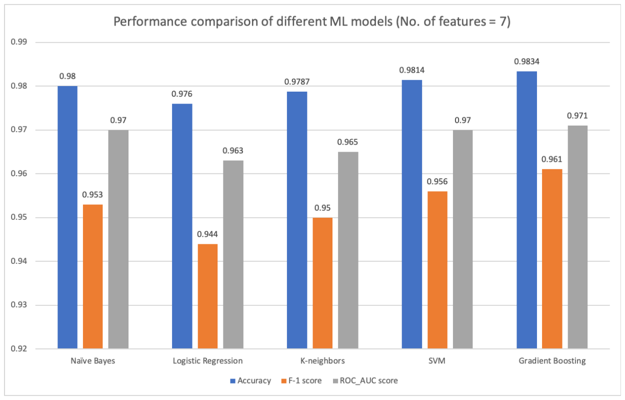


Fig. 5. Comparison of different classifiers (No. of features selected = 7)

classifiers was 7. For 9 features, Naive Bayes gave us the worst classifier in terms of all metrics. Depending on the value of k, K-nearest neighbor classifiers achieved inconsistent accuracy. The optimal value of k was found to be 3. In comparison with other classifiers, Logistic Regression didn't perform well when the number of features was reduced to 7. Considering all the metrics together, SVM and Gradient Boosting classifier proved to be better and consistent under various circumstances. Both models have an accuracy of over 98%, F1-score of over 95%, and ROC AUC score is over 97%. Gradient Boosting edged it over SVM by a little margin. Being an ensemble learner, Gradient Boosting combines weak learners and slowly learns from the error it gets in each iteration. We have used a slow learning rate of 0.2 to avoid overfitting problem, one of the most common problems in ensemble learning methods [20].

The area under the curve score is proved to be most useful for binary classification [16]. The no-skill ROC AUC score for a binary classification is 0.5 which is the diagonal line that runs from 0.0 to 0.1 for both TPR and FPR. Fig. 6 shows the comparison of all classifier's AUC score. The best classifier in our comparative study in terms of AUC score is Gradient Boosting.

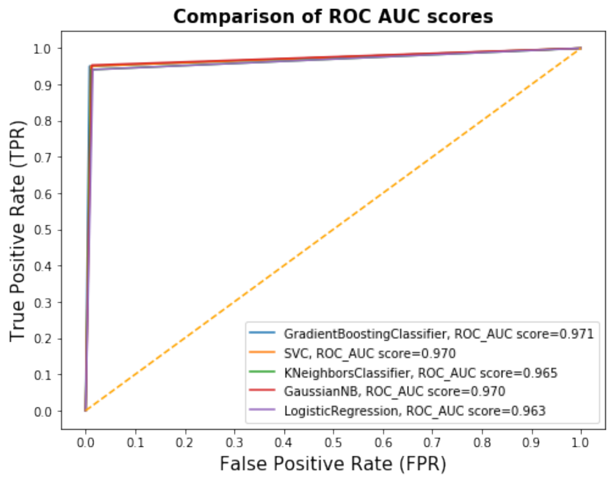


Fig. 6. ROC AUC score Comparison

V. Conclusion and Future Work

In this paper, we presented a model that applies SVM, Gradient Boosting, Gaussian Naive Bayes, k-NN, and Logistic Regression to prevent blackhole attacks in a VANET environment. We implemented our scheme in the NS-3 simulator under IEEE 802.11p WAVE standards with AODV routing protocol. We used data sets consisting of traffic flows with malicious and non-malicious vehicles. To check the validity of our model, we used accuracy, F-1 score, and ROC AUC as the performance metrics. Simulation results demonstrate that the ROC AUC scores for the five different algorithms are as follows: Logistic Regression (96.3%), k-NN (96.5%), Gaussian Naive Bayes (97%), SVC (97%), and Gradient Boosting (97.1%). On average, the ROC AUC score for the five algorithms is 96.78% which is much higher than a no skill ROC AUC score and only 3.22% away from an ideal ROC AUC score. Our scheme is robust and accurate as evidenced by its ability to identify and prevent blackhole attacks. Moreover, the scheme is scalable in that addition of vehicles to the network does not compromise its accuracy and robustness. For future work, we would like to test the model under Nakagami and Rayleigh fading protocols.

References

[1] Pranav Kumar Singh, Rahul Raj Gupta, Sunit Kumar Nandi, and Sukumar Nandi. "Machine learning based approach to detect wormhole attack in VANETs". In: *Workshops of the international conference on advanced information networking and applications*. Springer. 2019, pp. 651–661.

[2] Benedikt Brecht and Thorsten Hehn. "A security credential management system for V2X communications". In: *Connected Vehicles*. Springer, 2019, pp. 83–115.

[3] John B Kenney. "Dedicated short-range communications (DSRC) standards in the United States". In: *Proceedings of the IEEE* 99.7 (2011), pp. 1162–1182.

[4] Yi Zeng, Meikang Qiu, Zhong Ming, and Meiqin Liu. "Senior2local: A machine learning based intrusion detection method for vanets". In: *International conference on smart computing and communication*. Springer. 2018, pp. 417–426.

[5] Liang Zhao, Yujie Li, Chao Meng, Changqing Gong, and Xiaochun Tang. "A SVM based routing scheme in VANETs". In: *2016 16th International Symposium on Communications and Information Technologies (ISCIT)*. IEEE. 2016, pp. 380–383.

[6] Steven So, Prinkle Sharma, and Jonathan Petit. "Integrating plausibility checks and machine learning for misbehavior detection in VANET". In: *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE. 2018, pp. 564–571.

[7] Jyoti Grover, Nitesh Kumar Prajapati, Vijay Laxmi, and Manoj Singh Gaur. "Machine learning approach for multiple misbehavior detection in VANET". In: *International conference on advances in computing and communications*. Springer. 2011, pp. 644–653.

[8] Jordan Montenegro, Cristhian Iza, and Mónica Aguilar Igartua. "Detection of Position Falsification Attacks in VANETs Applying Trust Model and Machine Learning". In: *Proceedings of the 17th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks*. 2020, pp. 9–16.

[9] FY Osisanwo, JET Akinsola, O Awodele, JO Hinmikaiye, O Olakanmi, and J Akinjobi. "Supervised machine learning algorithms: classification and comparison". In: *International Journal of Computer Trends and Technology (IJCTT)* 48.3 (2017), pp. 128–138.

[10] Stephen Marsland. *Machine learning: an algorithmic perspective*. CRC press, 2015.

[11] George F Riley and Thomas R Henderson. "The ns-3 network simulator". In: *Modeling and tools for network simulation*. Springer, 2010, pp. 15–34.

[12] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[13] Kaitlin Kirasich, Trace Smith, and Bivin Sadler. "Random Forest vs logistic regression: binary classification for heterogeneous datasets". In: *SMU Data Science Review* 1.3 (2018), p. 9.

[14] Barkouk Hamid and En-Naimi El Mokhtar. "Performance analysis of the Vehicular Ad hoc Networks (VANET) routing protocols AODV, DSDV and OLSR". In: *2015 5th International Conference on Information & Communication Technology and Accessibility (ICTA)*. IEEE. 2015, pp. 1–6.

[15] Gustavo Carneiro, Pedro Fortuna, and Manuel Ricardo. "Flowmonitor: a network monitoring framework for the network simulator 3 (ns-3)". In: *Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools*. 2009, pp. 1–10.

[16] Andrew P Bradley. "The use of the area under the ROC curve in the evaluation of machine learning algorithms". In: *Pattern recognition* 30.7 (1997), pp. 1145–1159.

[17] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585 (2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2.

[18] The pandas development team. *pandas-dev/pandas: Pandas*. Version latest. Feb. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

[19] Executable Books Community. *Jupyter Book*. Version v0.10. Feb. 2020. DOI: 10.5281/zenodo.4539666. URL: https://doi.org/10.5281/zenodo.4539666.

[20] Candice Bentéjac, Anna Csörgő, and Gonzalo Martınez-Muñoz. "A comparative analysis of gradient boosting algorithms". In: *Artificial Intelligence Review* (2020), pp. 1–31.