Victor de Freitas Arruda

UM ESTUDO COMPARATIVO DE ALGORITMOS BASEADOS EM APRENDIZADO DE MÁQUINA PARA CLASSIFICAÇÃO DE TRÁFEGO EM REDES DEFINIDAS POR SOFTWARE

Montes Claros - MG

Dezembro de 2023

Victor de Freitas Arruda

UM ESTUDO COMPARATIVO DE ALGORITMOS BASEADOS EM APRENDIZADO DE MÁQUINA PARA CLASSIFICAÇÃO DE TRÁFEGO EM REDES DEFINIDAS POR SOFTWARE

Dissertação apresentada ao Mestrado Profissional em Modelagem Computacional e Sistemas, da Universidade Estadual de Montes Claros, como exigência para obtenção do grau de Mestre em Modelagem Computacional e Sistemas.

Orientador: Dr. Nilton Alves Maia
Coorientador: Dr. Maurílio José Inácio

Montes Claros - MG

Dezembro de 2023

Victor de Freitas Arruda

UM ESTUDO COMPARATIVO DE ALGORITMOS BASEADOS EM APRENDIZADO DE MÁQUINA PARA CLASSIFICAÇÃO DE TRÁFEGO EM REDES DEFINIDAS POR SOFTWARE/ Victor de Freitas Arruda. – Montes Claros - MG, Dezembro de 2023-

30 p. : il. (algumas color.) ; 30 cm.

Orientador: Dr. Nilton Alves Maia

Dissertação (Mestrado) – Universidade Estadual de Montes Claros – Unimontes
Centro de Ciências Exatas e Tecnológica
Programa de Pós-Graduação em Modelagem Computacional e Sistemas, Dezembro de 2023.

1. Redes Definidas por Software. 2. Machine Learning. 3. Classificação de Tráfego. I. Maia, Nilton Alves. II. Inácio, Maurílio José. III. Universidade Estadual de Montes Claros. IV. Programa de Pós-Graduação em Modelagem Computacional e Sistemas. V. Classificação de Tráfego em Redes Definidas por Software.

UNIVERSIDADE ESTADUAL DE MONTES CLAROS
PRÓ-REITORIA DE PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM COMPUTACIONAL E SISTEMAS

FOLHA DE APROVAÇÃO

Um estudo comparativo de algoritmos baseados em Aprendizado de Máquina para classificação de Tráfego em Redes Definidas por Software

Victor de Freitas Arruda

Trabalho de conclusão defendido e aprovado pela banca examinadora constituída por:

[illegible]

Dr. Nilton Alves Maia – Orientador
Departamento de Ciênciasda Computação – UNIMONTES

Maurílio José Inácio

Dr. Maurílio José Inácio – Coorientador
Departamento de Ciências Exatas – UNIMONTES

[illegible]

Dr. Allysson Steve Mota Lacerda
Departamento de Ciências da Computação – UNIMONTES

[illegible]

Dr. Marcel Veloso Campos
Departamento de Ciências Exatas – UNIMONTES

Romerson Deiny Oliveira

Dr. Rômerson Deiny Oliveira
University of Bristol (UK)

Montes Claros, 04 de Dezembro de 2023

Agradecimentos

Em primeiro lugar, gostaria de agradecer à Deus, que sempre me ajuda a alcançar meus objetivos de buscar aprender cada vez mais.

Agradeço também ao meu orientador Dr. Nilton Alves Maia e ao meu coorientador Dr. Maurílio José Inácio, pela oportunidade, direcionamento durante o desenvolvimento desta pesquisa, dando apoio durante todo o processo de construção desse trabalho.

Aos meus professores do PPGMCS pelas correções e ensinamentos que me permitiram apresentar um melhor desempenho no meu processo de formação profissional ao longo do curso.

Agradeço aos meus pais Maria Lúcia de Freitas Arruda e Sebastião Pereira de Arruda, que me deram apoio e incentivo nas horas difíceis, força e amor incondicional.

À Universidade Estadual de Montes Claros, juntamente com o PPGMCS, pela oportunidade de cursar o mestrado e por ter oferecido um ambiente criativo e amigável.

Resumo

Atualmente a internet tornou-se uma ferramenta muito importante para a troca de informações, fazendo com que ela necessite de um bom funcionamento. Nesse contexto, a adoção de Redes Definidas por Software viabiliza o desenvolvimento de novas técnicas para aprimorar o gerenciamento e desempenho das redes atuais. Como as redes definidas por Software separam o plano de dados do plano de controle, a lógica de encaminhamento é centralizada em um ou mais controladores, os quais possuem visão de grande parte da rede, ou até mesmo global quando são utilizados em conjunto. Com base na classificação do fluxo de tráfego entrante da rede pode-se implementar políticas de segurança, controle de qualidade de serviço e engenharia de tráfego, visando sempre melhorar o gerenciamento da rede. Este trabalho apresenta três artigos, os quais apresentam um estudo comparativo de algoritmos de aprendizado de máquina utilizando-se principalmente as métricas convencionais (acurácia, precisão, revocação e f1-score), onde verifica-se o desempenho de algoritmos de aprendizado de máquina em duas topologias propostas.

Palavras-chave: Redes Definidas por Software; Aprendizado de Máquina; Gerenciamento de redes; Qualidade de Serviço; Engenharia de Tráfego.

Abstract

Currently, the internet has become a very important tool for exchanging information, making it necessary for it to function properly. In this context, the adoption of Software Defined Networks enables the development of new techniques to improve the management and performance of current networks. As Software defined networks separate the data plane from the control plane, the forwarding logic is centralized in one or more controllers, which have a view of a large part of the network, or even a global view when used together. Based on the classification of the network's incoming traffic flow, security policies, service quality control and traffic engineering can be implemented, always aiming to improve network management. This work presents three articles, which present a comparative study of machine learning algorithms using mainly conventional metrics (accuracy, precision, recall and f1-score), where the performance of machine learning algorithms is verified in two proposed topologies.

Keywords: Software Defined Networks; Machine Learning; Traffic Engineering; Quality of Service; Networks Management.

Sumário

1 INTRODUÇÃO 8

2 ARTIGO IJER EM INGLÊS 11

3 CONSIDERAÇÕES FINAIS 30

1 Introdução

Após o surgimento da internet, houve uma revolução na comunicação e acesso à informação, tornando as redes de computadores muito importantes para o dia a dia. Sendo elas utilizadas das mais diversas formas e por praticamente todas as pessoas, seja direta ou indiretamente, a internet está sempre lá e estamos sempre conectados.

Assim torna-se importante que as redes de computadores estejam funcionando de forma adequada, pois estão sujeitas a diversos fatores, como por exemplo, ataques maliciosos, interrupção de serviços, sobrecargas, e outros fatores. Ademais, atualmente a configuração de regras na rede é individual para cada dispositivo, na qual os usuários utilizam comandos de gerenciamento que são específicos de cada fabricante, o que torna difícil a evolução de tecnologias utilizadas nas redes de computadores.

Nos dias de hoje, a quantidade de aplicações que consomem recursos da rede está aumentando, o que produz escassez de recursos de rede e faz com que haja a busca pelo desenvolvimento de novas tecnologias que melhorem o desempenho da rede. Dessa forma, a identificação dos fluxos de tráfego entrante é de grande importância para o gerenciamento, controle de tráfego e detecção de anomalias na rede.

Nesse cenário, as Redes Definidas por Software podem facilitar o desenvolvimento de técnicas que melhoram a qualidade de serviço da rede e enfrentar melhor os problemas de gerenciamento que ocorrem nas redes IP atuais. As redes definidas por software são caracterizadas pela separação do plano de dados e de controle, sendo que a lógica de encaminhamento de pacotes é centralizada no plano de controle (controlador). Assim, as alterações na lógica de encaminhamento são realizadas apenas no controlador que possui uma visão de grande parte da rede, ou até mesmo global quando os controladores são utilizados em conjunto.

Logo, a classificação de tráfego da rede, que visa classificar os fluxos de tráfego por seus aplicativos de geração, pode desempenhar um papel importante na segurança e gerenciamento da rede, como por exemplo, no controle de qualidade de serviço (QoS), interceptação legal, detecção de intrusão, sobrecargas e ataques maliciosos. A classificação de tráfego e a caracterização de aplicações de rede estão se tornando cada vez mais importantes para aplicações de engenharia de tráfego, monitoramento de segurança e qualidade de serviço.

Nesse contexto, durante a realização deste trabalho foram realizados três artigos, os quais apresentaram estudos em classificação de tráfego em redes definidas por software.

O primeiro artigo denominado "Classificação de Tráfego em Redes Definidas por

Software utilizando SVM e MLP: Um Estudo Comparativo"foi apresentado no **XXIV ENMC - Encontro Nacional de Modelagem Computacional** e publicado em seus anais.

O primeiro artigo teve como objetivo realizar um estudo comparativo dos algoritmos SVM e MLP usando ADAM e LBFGS para classificação de tráfego entrante em uma rede SDN. Os desempenhos dos classificadores foram avaliados considerando-se os valores da acurácia, precisão, revocação e F1-score. Verificou-se que o SVM obteve os melhores resultados.

O segundo artigo, denominado "Classificação de tráfego entrante em uma topologia SDN", foi publicado em 2021 na revista **Cereus** (ISSN: 2175-7275, Edição: v.13 n.3) e está disponível no link: http://ojs.unirg.edu.br/index.php/1/article/view/3426.

No segundo artigo foi proposto um estudo comparativo de algoritmos de aprendizado de máquina para a classificação do tráfego entrante em uma topologia SDN. O desempenho dos classificadores foi avaliado através das métricas acurácia, precisão, revocação e f1-score, além dos tempos de treinamento e validação dos modelos. O algoritmo Random Forest foi considerado o mais eficiente no cenário de classificação de tráfego considerado. Ele alcançou valores semelhantes aos melhores resultados com relação às métricas acurácia, precisão, revocação e f1-score, mas obteve valores inferiores nos tempos de treinamento e validação.

O terceiro artigo, denominado "A comparative study of Machine Learning-Based Algorithms for traffic classification in Software-defined Networks"é apresentado em Inglês no capítulo 2 desse documento. O artigo foi submetido e aceito na revista **International Journal of Education and Research (IJER)** (ISSN 2411-5681).

No terceiro artigo foi realizado um estudo comparativo em duas topologias SDN diferentes utilizando os algoritmos de redes neurais artificiais do tipo Multilayer Perceptron (utilizando o ADAM e o LBFGS), Máquinas de Vetores de Suporte, NaiveBayes, K-NearestNeighbor, Random Forest e o Ensemble (votação). A análise estatística mostrou que na topologia 1, o Random Forest se destacou nas métricas convencionais acurácia, precisão, revocação, F1-score. Na topologia 2, o Random Forest obteve melhor Acurácia e Revocação, a RNA do tipo MLP com o ADAM obteve melhor Precisão e o SVM obteve melhor F1-Score. No tempo de treinamento do modelo, o KNN e o NaiveBayes obtiveram resultados semelhantes nas duas topologias. Em relação aos melhores tempos de validação do modelo, o NaiveBayes obteve os melhores resultados nas duas topologias. Porém, como o NaiveBayes obteve resultados comparativamente inferiores, principalmente na topologia 2, em relação as métricas convencionais, conclui-se que o Random Forest foi o algoritmo com melhor resultado geral neste trabalho.

Nos três trabalhos realizados observou-se a possibilidade da classificação de tráfego entrante em topologias de redes definidas por software, sendo que o Random Forest foi o algoritmo que mais se destacou.

2 Artigo IJER em inglês

A COMPARATIVE STUDY OF MACHINE LEARNING-BASED ALGORITHMS FOR TRAFFIC CLASSIFICATION IN SOFTWARE-DEFINED NETWORKS

ABSTRACT

Computer networks have become a vital tool for information transportation. Use of Software-Defined Networks (SDN) can enable the development of techniques to enhance network performance in terms of security, quality of service, and traffic engineering. The implementation of these techniques can be facilitated by classifying incoming network traffic. This study conducted a comparative analysis on two different SDN topologies using artificial neural network algorithms, namely Multilayer Perceptron (using ADAM and LBFGS), Support Vector Machines, Naive Bayes, K-Nearest Neighbors, Random Forest, and Ensemble (voting). The statistical analysis revealed that in topology 1, Random Forest stood out in the conventional metrics of accuracy, precision, recall, and F1-score. In topology 2, Random Forest achieved better accuracy and recall, MLP with ADAM had better precision, and SVM had better F1-score. Regarding model training time, KNN and Naive Bayes produced similar results in both topologies. As for the best model validation times, Naive Bayes yielded the best results in both topologies. However, since Naive Bayes showed comparatively inferior results, especially in topology 2, in terms of conventional metrics, it is concluded that Random Forest was the algorithm with the best overall performance in this study.

Key-words: Machine Learning, Software Defined Networks, Traffic Classification.

1.0 INTRODUCTION

With the creation of computer networks, there was a revolution in access to information, making it very important in everyday life. They are used as support in the most varied activities and have become a vital tool for the transport of information in today's world.

Therefore, networks must function properly as they are susceptible to various factors such as overload, service interruptions, and malicious attacks, among others. Additionally, developing new technologies within traditional IP networks is challenging. These networks consist of multiple devices, typically comprised of proprietary software from specific manufacturers, running on proprietary hardware. This limits the creation and development of new technologies for current networks, making them inflexible (Cardoso, Silva, Rocha, and Sousa, 2017).

In present days, there has been an increase in the number of applications that consume traffic, which ends up producing a scarcity of network resources. This makes it necessary to search for the development of new technologies that improve network performance. Thus, the identification of incoming traffic flows is of great importance for the management, traffic control, and detection of anomalies in the network (Ding, Yu, Peng, and Xu, 2013).

In this scenario, Software Defined Network (SDN) can facilitate the development of techniques to improve the network quality of service (QoS). The SDN is characterized by the separation of the data plane and the control plane, and the packet forwarding logic is centralized in the control plane (controller). Therefore, changes in the forwarding logic are performed only in the controller that has a view of a large part of the network, or even a global one (Bisol, Silva, Machado, Granville, and Schaeffer-Filho, 2016).

The SDN networks are a new paradigm in telecommunications and computer networks. They are intended to help resolve management issues in today's IP networks. As network administrators are responsible for configuring and applying high-level policies to a wide range of events that may occur, SDN networks offer hope for using more convenient methods for network configuration and management. The SDN architecture is divided into three layers, with the lowest level being the data plane, the intermediate level being the control plane and the highest level being the application plane. The data plane simply acts as packet forwarding hardware, it communicates with the control plane through the southbound interface. This interface enables the communication between the programmable switches (SDN switches) and the controller, that is, the controller uses a southbound interface from the SDN-enabled switch devices to connect to the data plane. The control plane acts as the "brain" of the network. The control plane is easily programmable and provides an abstraction of the underlying network infrastructure. This allows switches to become simpler devices as they accept instructions from the centralized controller. In this manner, the network administrator does not need to configure the network devices individually and the routing and forwarding decisions are implemented through the centralized SDN controller (Kokila, Selvi, and Govindarajan, 2014). The communication between the network applications and the controllers is maintained by the northbound interface, located in the control plane. The northbound interface determines how to express operational tasks and network policies, and also how to convert them into a format the controller can understand (Kim and Feamster, 2013) (Farhady, Lee, and Nakao, 2015).

The separation of data and control planes, provided by SDN networks, enables the development of applications aimed at improving network management. In this manner, it is possible to develop, for example, applications to solve security, quality of service (QoS), and traffic engineering problems. These problems can be better solved if it is possible to obtain more precise information about the incoming traffic on the network, which can be obtained from its classification.

According (to Bakker, Ng, Seah, and Pekar, 2019), traffic classification describes the process of identification and the pairing packet flows for some type of traffic (malicious traffic, low-priority traffic, network applications, etc.). This process depends on information extracted from the traffic that serves as input to the traffic classification algorithm. The purpose of traffic classification is to improve network resource management, network security, and QoS. The precise traffic classification, done promptly is becoming more and more important for many wired and wireless networking applications such as traffic engineering, security monitoring, and QoS (Fan and Liu, 2017). The traffic classification can be performed using port numbers, payload inspection, and machine learning techniques.

The traffic classification by port number relies only on mapping applications to identify known port numbers. Unfortunately, over time, traffic classification techniques based on port numbers have become imprecise and some limitations have become obvious. A large number of applications appeared that did not have registered port numbers and many of them used dynamic port negotiation mechanisms to hide from firewalls and network security tools (Fan and Liu, 2017) (Amaral, et al., 2016).

The classification based on payload inspection, also known as Deep Packet Inspection (DPI), is currently one of the most widely used techniques (Amaral, et al., 2016). The DPI systems use the predefined signatures of packets or flows to describe which traffic type the packet or flow belongs to. In this way, DPI systems inspect the payload of packets of a given flow to match the packet or flow with predefined signatures, and the matching process is always done by regular expressions. Like payload-based methods need to examine the payload of every packet, sometimes some issues such as privacy laws and cryptography that can lead to an inaccessible traffic payload. On the other side, DPI results in high computational costs and requires manual maintenance of signatures (Fan and Liu, 2017).

The classification methods based on machine learning can overcome some of the limitations of port and payload-based approaches. More specifically, machine learning techniques can classify Internet traffic using application protocol-independent statistics, like flow duration, packet length variance, maximum or minimum segment size, window size, round trip time, and packet arrival time. Thus, it can lead to lower computational costs and identify encrypted traffic easily (Fan and Liu, 2017). The traffic classification can be performed using supervised or unsupervised learning. In supervised learning, is necessary to obtain labeled training datasets on which new applications can appear. Unsupervised learning is typically used for clustering tasks, where algorithms group data into different clusters according to similarities in feature values. The goal is to identify unknown relationships in the data by finding patterns of similarity between many observations (Amaral, et al., 2016). The classifiers used in this work use supervised learning. In particular, were used, Ensemble (voting), Random Forest, Support Vector Machine (SVM), Naive Bayes, K-Nearest Neighbor (KNN), and artificial neural networks (ANNs) type Multi-Layer Perceptron (MLP) using ADAM and Limited-Memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS).

The ANNs of type MLP allow for solving complex problems, that are not possible to be solved by the basic model of neurons. The internal neurons of MLP are very important in the ANN, because it is proven that without them it is impossible to solve linearly non-separable problems (Ferreira, 2004). In this type of network, every layer has its specific function. Thus, each neuron calculates a weighted sum of the inputs and passes this sum in the form of a bounded nonlinear function. At the mesostructure level, there are two or more layers with a feedforward connection (Vieira and Bauchspiess, 2016). It can be said that with the addition of one or more intermediate (hidden) layers, the computational power of non-linear processing and storage of the network is increased. In a large enough hidden layer, it is possible to represent any continuous function of the inputs. The outputs of the neurons of each intermediate layer are used as input for the posterior layer

The Support Vector Machine (SVM) was developed by Vapnik (1995), from the studies of Vapnik and Chervonenkis (1971) and Boser, Guyon, and Vapnik (1992). Classification through SVM consists of the optimal separation of a group of data, regardless of its dimensionality, through a quadratic programming problem that allows a good generalization (Vapnik, 1998). This process causes the SVM to find a global minimum on the cost surface, which is considered an advantage of the method (Haykin, 2001).

Naive Bayes is a machine learning algorithm, where the classifier learns through a document classification algorithm (Bužic and Dobša, 2018). The Naive Bayes classifier is based on two basic hypotheses: 1) the characteristics are independent of each other, and 2) each feature has the same prominence (Wu, et al., 2018). The Naive Bayes classifier takes an arbitrary number of continuous or categorical variables and classifies an instance to belong to one of several classes. Therefore, it is applied to learning tasks where each instance x is described by a conjunction of attribute values and the target function f(x) (Vaidya, Shafiq, Basu, and Hong, 2013). This classifier is based on the Bayes theorem. The Efficiency in modeling and prediction is an undoubted advantage over other classification algorithms, which is due to the possibility of easy parallelization, especially important for large datasets (Bužic and Dobša, 2018).

The K-Nearest Neighbor (KNN) is an algorithm that uses clustering to classify data, in this way, it can use supervised learning and unsupervised learning to perform classification, in the context of this work supervised learning was used it requires training data and a predefined k value to find the k closest data based on distance calculation (Chomboon, Chujai, Teerarassammee, Kerdprasop, and Kerdprasop, 2015). The KNN method, although simple, generally can match and even outperform more sophisticated and complex methods in terms of generalization error. One of the biggest problems with this classifier, however, is to fix the appropriate value of k (García-Pedrajas, Romero del Castillo, and Cerruela-García, 2017). The KNN builds predictions directly from the training dataset, which is stored in memory. To classify unknown data, for example, KNN finds the set of k training data objects closest to the input data instance by a distance calculation and assigns the maximum voted classes from these neighboring classes (Singh, Halgamuge, and Lakshmiganthan, 2017).

The Random Forest classifier uses multiple decision trees during the training phase and produces the average prediction of individual trees (Edla, Mangalorekar, Dhavalikar, and

Dodia, 2018). To predict the target value for a new instance of data, the new observation is fed through all the classification trees in the Random Forest. The prediction numbers for a class realized by each of the classification trees are counted. Then the class with the maximum number of votes is returned as the class label for the new neighboring data instance (Singh, Halgamuge, and Lakshmiganthan, 2017). Some properties that make Random Forest a very good classification model for large datasets, as (a) no need to prune trees, (b) automatic generation of precision and variable importance, (c) not being very sensitive to outliers in the training data, and (d) be an easy model to define parameters (Jedari, Wu, Rashidzadeh, and Saif, 2015).

The Ensemble methods combine predictions from several basic classifiers and provide the final prediction. The ensemble model combines a set of classifiers to create a single composite model that provides better accuracy. Ensemble methods can be defined as committee, classifier fusion, combination or aggregation, voting, etc. Many researches show that forecasting a composite model provides better results compared to forecasting a single model (Gandhi and Pandey, 2015). The ensemble method performance depends on the accuracy of the individual classifiers and the number of base classifiers included (that is, the more classifiers we include, the better the performance of the ensemble classifier) (Saqlain, Jargalsaikhan, and Lee, 2019). Ensemble methods can be classified as homogeneous or heterogeneous Ensemble methods. Homogeneous Ensemble methods use a single learning algorithm on different training datasets to build multiple classifiers such as Bagging, Boosting, Random Subspaces, and Random Forest, etc. Heterogeneous Ensemble methods use various machine learning algorithms and manipulate training datasets to make various models. Some of the heterogeneous methods are voting, stacking, etc. (Gandhi and Pandey, 2015). In this work, we used the voting heterogeneous ensemble method. In the voting method, each base classifier's prediction is counted as a vote for a class, and the final class assigned is the one that garners the most votes (Badhani and Muttoo, 2019).

In order to verify whether there is a significant difference between the results of the classifiers used, Friedman's test and Conover's post-hoc test were applied. The Friedman test is non-parametric, that is, it does not assume a distribution from the performance indicators, so this test is used to verify whether the analyzed data are statistically similar. Therefore, the null hypothesis (H0) was proposed, which assumes that the distributions of the k samples are identical, and the Alternative Hypothesis (H1), which states that the distributions of the k samples differ in location (Firmino, 2015). If the null hypothesis is rejected, it is possible to use post-hoc tests to verify which of the analyzed groups are different from each other. In this case, Conover's post-hoc test was used, which is capable of finding statistically significant differences between the results of the classifiers.

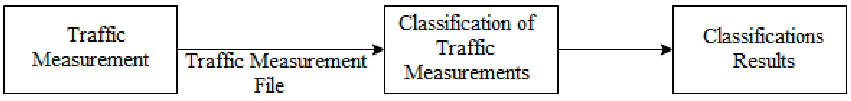
This work proposes the implementation of an incoming applications traffic classification system in two SDN topologies. The applications are VoIP, Telnet, Quake3, DNS, CSi, CSa, Video and Web. To perform the classification, SVM, KNN, Naive Bayes, Random Forest and Ensemble, and MLP-type ANN models (using ADAM and LBFGS) were compared.

The next sections of this article are organized as follows: in section 2 the methodology is presented. Section 3 presents the results obtained from the experiments. In section 4 the results obtained are discussed, and then in section 5, the conclusions about the work are presented.

2.0 METHODOLOGY

Initially, traffic is measured in the SDN topology. The data resulting from the traffic measurement is stored in a file. These data are used by machine learning algorithms to perform classifications. A summary of the methodology used in this work is shown in Figure 1.

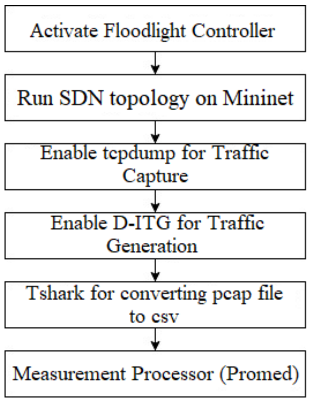
Figure 1- Methodology used in this work



The simulation of the SDN topology was performed using the Mininet software (Mininet Team, 2021) running on the SND Hub platform (SDN Hub Team, 2021). The code for implementing SDN topologies on Mininet was written in Python language. The software D-ITG (Distributed Internet Traffic Generator) (Botta, Dainotti, and Pescapé, 2012) software was used to generate application traffic from hosts. To capture the traffic, tcpdump was used. It is a libpcap-based, API to capture network packets during their traffic. With the traffic captured, tcpdump generates a file in pcap format, this file is transformed into csv by tshark, which is an implementation of Wireshark in text mode, it is very similar to tcpdump, but it allows the conversion of the pcap file to csv (which is used in this work).

The steps used to perform the traffic measurement are described in Figure 2. First, the floodlight controller is started. Then the SDN topology is started in the Mininet software. The tcpdump application starts to capture the traffic passing through the network interfaces. With the SDN topology working, D-ITG starts to generate traffic on the network. As tcpdump doesn't convert the pcap file to csv, tshark is used to convert the file to csv type. The promed (measurement program) processes the csv file and generates the traffic measurement.

Figure 2- Steps for performing traffic measurement


Activate Floodlight Controller
Run SDN topology on Mininet
Enable tcpdump for Traffic Capture
Enable D-ITG for Traffic Generation
Tshark for converting pcap file to csv
Measurement Processor (Promed)

In the SDN network topologies considered in this work, the hosts were configured as receivers and traffic generators. Every generator host transmits traffic packets from a different network application. The selected network applications were Voip, Telnet, Quake3, DNS, CSa (Counter Strike Active), CSi (Counter Strike Inactive), Web, and Video. VoIP is a voice over IP application. Telnet is an application that allows remote access to any machine running the server module. Quake3 is a first-person shooter video game. DNS simulates an application that is responsible for locating and translating website addresses typed into browsers into IP numbers. CSa and CSi are first-person shooter games that simulate when the user is active and inactive respectively. Web and Video simulate web and video applications respectively.

In the promed was used the pandas library (Pandas Development Team, 2021) and the spark (Commiters, 2021) for processing large amounts of data. The measurement generated was done by applying metrics to the information present in the packet headers. At the end of the processing, a new file was generated and this measurement file is used as input in the classification algorithms.

The SDN topology was in operation for 1 hour, so each traffic measurement lasted 10 seconds, and in the end, 360 measurement patterns were obtained. The three hundred and sixty measurement standards for each application were separated into two sets of data, with seventy-five percent (75%) being used for algorithm training and the other twenty-five percent (25%) for validation.

The Attributes of measurement patterns used by machine learning algorithms during training and validation were average delay, average jitter variation, throughput, average packet rate per second, number of transmitted packets, and number of transmitted bytes. The outputs of the algorithms are known network applications (traffic measurement classes). The classifiers were selected based on other existing works in the literature. All machine learning algorithms were implemented using the Python language and the library Scikit-learn (Pedregosa, et al., 2011).

The MLP network with ADAM was configured with three (3) hidden layers with one hundred (100), fifty (50), and fifty (50) neurons, respectively. Furthermore, the model used the hyperbolic tangent activation function (tanh) and the penalty parameter (alpha) equal to 0.0001. The MLP network with LBFGS was configured with two (2) hidden layers with one hundred (100) neurons each. The model used the logistic activation function (logistic) and the penalty parameter (alpha) with a value of 0.05. In both models, the type of learning rate invscaling was used and the maximum number of interactions equal to five hundred (500).

For the SVM algorithm, the complexity (C) was defined as one thousand (1000), the relevance of the data closest to the separation frontier (gamma) was defined as one (1) and the kernel used was the RBF (Radial Basis Function).

In Random Forest ten (10) trees in the forest (n_estimators) were used. The minimum number of samples needed to split an internal node (min_samples_split) was set to ten (10). The maximum depth of the tree (max_depth) was set to null and the number of features considered when looking for the best split (max_features) was the square root of the number of features.

In KNN, the number of selected neighbors (n_neighbors) was defined as three (3), and for the distance (p) Manhattan was selected. The algorithm used to compute the nearest neighbor (algorithm) was defined as "auto" and the weights (weights) function defined in the prediction was the inverse of the distance.

In Naive Bayes, the largest variance of all features that are added to variances for calculation stability (var_smoothing) was set to 1.519911e-06.

In the case of the Ensemble, the Voting Ensemble was adopted, which used as a parameter of estimators for each of the algorithms mentioned above, with the best parameters found during cross-validation.

3.0 RESULTS

In this section, the results of the evaluation of the selected algorithms for classifying incoming traffic in the SDN topology are presented. Two different network topologies adapted from the work of (Maia, 2006) were used.

The first network topology used is topology 1, shown in Figure 3. It is formed by forty-eight (48) hosts, twenty-one (21) switches, and one (1) controller. The hosts were divided between clients and servers (receivers), being twenty-four (24) clients and twenty-four (24) servers. The bandwidth of the links between hosts and switches was set to 100Mbps.

The second network topology used is topology 2, shown in Figure 4. It is formed by fifty (50) hosts, sixteen (16) switches, and one (1) controller. The hosts were divided between clients and servers (receivers), being twenty-five (25) clients and twenty-five (25) servers. The bandwidth of the links between hosts and switches was set to 100Mbps.

To carry out the analysis, a hypothesis test was used, and two hypotheses were proposed. The null hypothesis (H0) says that the medians of the results of the algorithms using the analyzed metric are all equal, that is, the results of all machine learning algorithms, analyzing the metric in question, are statistically the same. Hypothesis H1 says that not all the medians of the results of the algorithms using the analyzed metric are equal, that is, at least the result of a machine learning algorithm, analyzing the metric in question, differs from the others.

Figure 3 - Topology 1 (Adapted from (Maia, 2006))

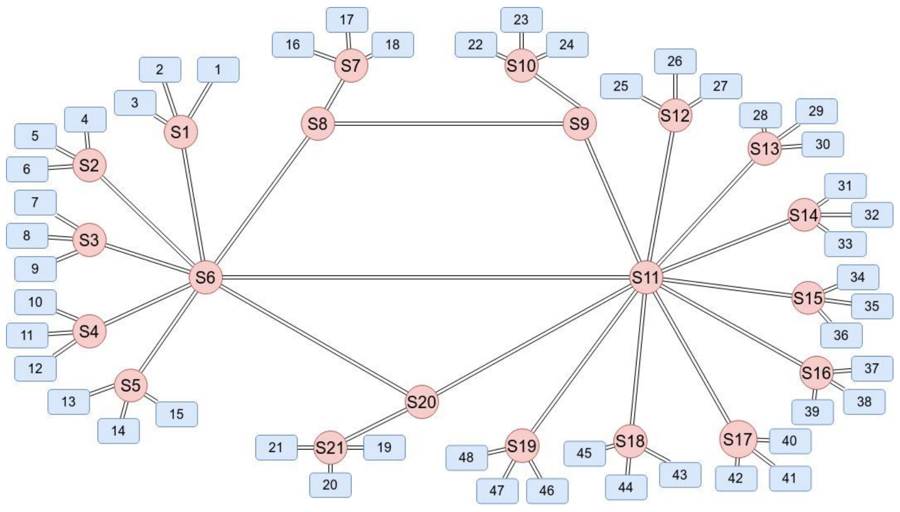
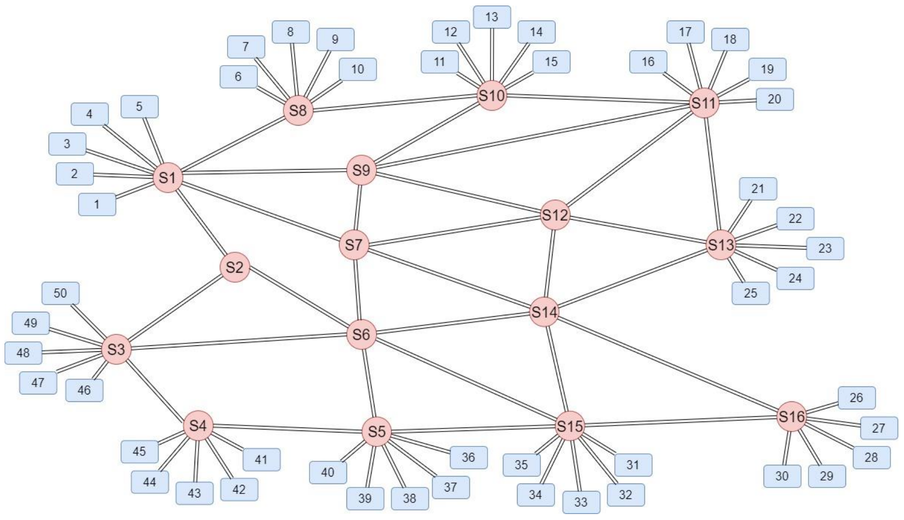


Figure 4 - Topology 2 (Adapted from (Maia, 2006))



After proposing the hypotheses, the Friedman test was applied to all selected metrics using 5% as a significance level and 6 as a degree of freedom. If the p-value found is less than the significance level, hypothesis H0 is rejected and the Conover post-hoc test is performed to verify which algorithms have different results.

Thus, hypotheses were made about the metrics Accuracy, Precision, Recall, F1-Score, model training time, and model validation time. Each of the classification algorithms was executed 100 (one hundred) times and the algorithms used the same parameters when processing the input data of the two topologies. For each execution of the algorithms, the input data were separated into training and validation data, and to carry out the statistical analysis each of the algorithms had to use the same set of data as the others in each execution.

Tables 1 and 2 present the results of the Conover test on the metrics used. If the p-value found is lower than the significance level divided by the number of comparisons (0.05/21=0.00238), it is said that the two compared algorithms had different results, otherwise they obtained similar results.

Table 1- Conover Test p-value result on the metrics used in Topology 1

Comparison	Accuracy	Precision	Recall	F1-Score	Training Time	Validation Time
adam – lbfgs	1,43905E-08	5,34054E-06	8,72234E-04	3,63345E-03	2,11032E-09	1,51543E-19
adam – svm	5,63385E-04	5,81806E-03	2,36003E-01	1,36650E-01	2,11032E-09	5,01569E-65
adam - random forest	3,59177E-04	1,52419E-25	5,58156E-02	9,75745E-01	1,45342E-30	6,70736E-05
adam - naive bayes	2,17318E-92	1,58540E-114	1,49921E-92	7,69300E-94	5,06263E-68	1,52419E-25
adam - k-nn	3,50428E-60	3,07783E-82	4,26390E-59	2,85772E-61	2,12517E-82	4,96544E-26
adam - ensemble	1,52556E-02	1,78423E-04	2,01930E-01	2,73966E-01	1,45342E-30	4,60385E-97
lbfgs - svm	5,27745E-19	6,85062E-02	7,06611E-06	1,22521E-05	1,45342E-30	8,91370E-22
lbfgs - random forest	1,94727E-19	4,10230E-10	1,99263E-07	3,99871E-03	2,08978E-59	6,87095E-36
lbfgs - naive bayes	3,31429E-62	2,71165E-90	7,34180E-75	2,20488E-78	4,02260E-100	4,41161E-70
lbfgs - k-nn	1,16503E-32	2,52814E-58	1,27433E-42	1,07879E-46	1,58540E-114	8,90323E-02
lbfgs - ensemble	9,70965E-04	4,11839E-01	3,90431E-02	6,85062E-02	2,11032E-09	1,81599E-48
svm - random forest	9,03205E-01	1,70342E-15	4,65683E-01	1,28838E-01	2,11032E-09	4,21052E-86
svm - naive bayes	1,05640E-110	5,82610E-100	7,79590E-99	9,91890E-102	5,89332E-38	1,35010E-122
svm - k-nn	3,18978E-78	1,04854E-67	3,49234E-65	5,68380E-69	5,24330E-51	8,59875E-16
svm - ensemble	6,08152E-09	3,15913E-01	1,40325E-02	9,96404E-03	2,08978E-59	2,11032E-09
random forest - naive bayes	2,42940E-111	4,32207E-57	1,07730E-102	1,11507E-93	3,82781E-14	1,06415E-11
random forest - k-nn	7,27979E-79	1,62071E-28	5,68380E-69	4,09008E-61	9,50817E-24	8,94891E-44
random forest - ensemble	3,01211E-09	2,11691E-12	1,47889E-03	2,87502E-01	2,92264E-91	3,55480E-118
naive bayes - k-nn	1,47411E-09	1,76442E-09	3,40713E-10	1,02665E-09	6,37685E-03	5,03098E-79
naive bayes - ensemble	1,66007E-79	1,20260E-94	8,84222E-86	4,90199E-88	8,0582E0-132	2,79940E-153
k-nn - ensemble	5,05914E-48	1,61439E-62	1,15060E-52	1,03915E-55	1,2773E-145	2,43526E-40

The selection of the best parameters for the selected algorithms in the scenario in question was done through Scikit-learn (Pedregosa, et al., 2011) using GridSearch cross-validation, where an exhaustive search was carried out on parameter values specified for each algorithm, presenting the parameters that obtained the best results. The data collected from topology 2 was used to perform crossvalidation, since they were obtaining the lowest values for the selected metrics.

Table 2 - Conover Test p-value result on the metrics used in Topology 2

Comparison	Accuracy	Precision	Recall	F1-Score	Training Time	Validation Time
adam – lbfgs	5,13730E-09	5,42888E-03	6,57408E-04	6,57408E-04	4,48296E-10	1,52690E-22
adam – svm	3,67725E-02	1,04516E-02	1,43316E-03	1,43316E-03	4,48296E-10	1,43537E-53
adam - random forest	1,11894E-40	1,05834E-08	2,50474E-10	2,50474E-10	8,47797E-33	2,15180E-08
adam - naive bayes	1,60145E-32	2,20898E-40	1,57245E-40	1,57245E-40	8,74387E-78	1,78150E-30
adam - k-nn	3,31790E-64	1,83165E-52	1,73237E-49	1,73237E-49	3,87765E-82	1,77983E-28
adam - ensemble	1,19674E-07	2,47160E-06	9,82089E-07	9,82089E-07	8,47797E-33	1,33369E-98
lbfgs - svm	1,37788E-04	8,24402E-01	8,24402E-01	8,24402E-01	8,47797E-33	1,22604E-11
lbfgs - random forest	1,54734E-16	2,69997E-03	2,69997E-03	2,69997E-03	2,17757E-63	1,78337E-47
lbfgs - naive bayes	6,01671E-61	6,92001E-54	3,09459E-57	3,09459E-57	2,62500E-111	1,25031E-80
lbfgs - k-nn	2,06954E-95	1,14942E-66	7,87038E-67	7,87038E-67	1,27120E-115	1,36631E-01
lbfgs - ensemble	5,68338E-01	4,97461E-02	1,28495E-01	1,28495E-01	4,48296E-10	6,10080E-46
svm - random forest	2,73988E-31	1,28559E-03	1,28559E-03	1,28559E-03	4,48296E-10	5,62264E-83
svm - naive bayes	5,15324E-42	8,85889E-53	4,08807E-56	4,08807E-56	1,75043E-45	4,12450E-117
svm - k-nn	4,09424E-75	1,62289E-65	1,11225E-65	1,11225E-65	1,73237E-49	8,54797E-08
svm - ensemble	1,15218E-03	2,90483E-02	8,16469E-02	8,16469E-02	2,17757E-63	5,80968E-17
random forest - naive bayes	1,22720E-105	2,64556E-69	8,75136E-73	8,75136E-73	1,33979E-18	2,05970E-10
random forest - k-nn	1,48630E-139	1,79129E-82	1,21742E-82	1,21742E-82	1,37885E-21	7,71310E-55
random forest - ensemble	1,73060E-18	2,95755E-01	1,36631E-01	1,36631E-01	9,37481E-97	2,87700E-128
naive bayes - k-nn	1,13989E-10	1,25026E-02	6,18340E-02	6,18340E-02	4,09977E-01	1,61263E-88
naive bayes - ensemble	4,86848E-58	7,04524E-64	5,03737E-65	5,03737E-65	7,56500E-144	1,67440E-160
k-nn - ensemble	2,19341E-92	5,98438E-77	8,81924E-75	8,81924E-75	5,99960E-148	6,48206E-39

3.1 ACCURACY ANALYSIS

Table 3 presents the results of the machine learning algorithms regarding the accuracy and the accuracy rank of the algorithms in topologies 1 and 2.

Table 3- Accuracy

	Rank (1)	Accuracy (1)	Rank (2)	Accuracy (2)
Random Forest	636,5	99,78%	562,5	91,35%
SVM	442,0	99,64%	560,5	91,33%
MLP(LBFGS)	502,5	99,68%	409,0	91,04%
Ensemble	493,5	99,68%	463,5	91,15%
MLP(ADAM)	409,0	99,49%	503,5	91,20%
KNN	106,5	98,85%	201,0	89,55%
Naive Bayes	210,0	99,24%	100,0	74,58%

When applying the Friedman test to the accuracies collected from the execution of the algorithms in topology 1, a p-value of 2.08086E-91 was obtained, thus rejecting the hypothesis H0. Therefore, through Conover's post-hoc test, presented in Table 1, it can be stated that the SVM algorithms and the ANN of type MLP using the ADAM algorithm have similar results. The Ensemble algorithms and the MLP using the LBFGS algorithm also showed similar results. The other algorithms obtained different results from each other.

The Random Forest (636.5) was the algorithm that stood out the most in the accuracy rank, referring to topology 1, shown in Table 3. Next, with similar results, are the ANN of type MLP using the LBFGS algorithm (502.5) and the Ensemble (493.5). The SVM algorithms (442) and the MLP using the ADAM algorithm (409) followed, which also obtained similar results. Finally, Naive Naive Bayes (210) and KNN (106.5).

When applying the Friedman test to the accuracies collected from the execution of the algorithms in topology 2, a p-value of 6.93952E-89 was obtained, thus rejecting hypothesis H0. Therefore, through Conover's post-hoc test, presented in Table 2, it can be stated that the SVM algorithms and the Random Forest have similar results. The MLP using the ADAM algorithm and Ensemble also showed similar results. The other algorithms obtained different results from each other.

The Random Forest (562.5) and SVM (560.5), with similar results, were the algorithms that stood out in the accuracy rank, referring to topology 2, shown in Table 3. Next, with similar results, we have the MLP using the ADAM algorithm (503.5) and the Ensemble (463.5). Finally, the MLP uses the LBFGS algorithm (409), KNN (201), and Naive Bayes (100).

3.2 PRECISION ANALYSIS

Table 4 presents the results of the machine learning algorithms regarding the precision and the precision rank of the algorithms in topologies 1 and 2.

Table 4- Precision

	Rank (1)	Precision (1)	Rank (2)	Precision (2)
Random Forest	526,0	100,00%	392,5	91,41%
SVM	475,0	99,85%	527,0	91,98%
MLP(LBFGS)	478,5	99,86%	497,0	91,86%
Ensemble	509,5	99,95%	510,5	91,92%
MLP(ADAM)	434,5	99,67%	572,5	92,19%
KNN	168,5	98,94%	200,5	89,64%
Naive Bayes	208,0	99,08%	100,0	68,21%

When applying the Friedman test to the precisions collected from the execution of the algorithms in topology 1, a p-value of 2.03667E-88 was obtained, thus rejecting the hypothesis H0. For this reason, through Conover's post-hoc test, we can state that the Random Forest algorithm is similar to Ensemble and the ANN of type MLP using the LBFGS algorithm. The Ensemble, in addition to Random Forest, is similar to the ANN of type MLP using the LBFGS algorithm and the SVM. The MLP using the LBFGS algorithm, in addition to Random Forest and Ensemble, is also similar to MLP using the ADAM algorithm and SVM. The SVM, in addition to the Ensemble and the MLP using the LBFGS, is also similar to the MLP using the ADAM algorithm. The MLP using the ADAM algorithm, as said, is similar to the MLP using the LBFGS algorithm and the SVM. Finally, Naive Bayes and KNN are similar to each other, the other algorithms had different results.

Random Forest (526) was the algorithm that stood out the most in the precision rank results, referring to topology 1, shown in Table 4. Posteriorly, with similar results, are the Ensemble (509.5), the MLP using the LBFGS algorithm (478.5), SVM (475.0), MLP using the ADAM algorithm (434.5). And the Naive Bayes (208) and the KNN (168.5).

When applying the Friedman test to the precisions collected from the execution of the algorithms in topology 2, a p-value of 6.8256E-104 was obtained, thus rejecting the hypothesis H0. Consequently, through Conover's post-hoc test, presented in Table 2, it can be stated that the ANN of type MLP using the ADAM algorithm obtained similar results to the SVM. SVM, in addition to MLP using ADAM algorithm, is similar to Ensemble and MLP using the LBFGS algorithm. The Ensemble, in addition to SVM, is similar to MLP using LBFGS. The MLP using

LBFGS algorithm, as mentioned, has similar results to SVM and Ensemble. The other algorithms obtained different results from each other.

The ANN of type MLP using the ADAM algorithm (572.5) was the algorithm that stood out the most in the precision rank results, referring to topology 2, presented in Table 4. Afterward, with similar results, are SVM (527), Ensemble (510.5), MLP using the LBFGS algorithm (497), and Random Forest (392.5). Finally, KNN (200.5) and Naive Bayes (100).

3.3 RECALL ANALYSIS

Table 5 presents the results of the machine learning algorithms regarding the recall and the recall rank of the algorithms in topologies 1 and 2.

When applying the Friedman test to recalls collected from the execution of the algorithms in topology 1, a p-value of 7.91246E-89 was obtained, thus rejecting the hypothesis H0. Consequently, through Conover's post-hoc test, presented in Table 1, it can be stated that Random Forest is similar to Ensemble and MLP using the LBFGS algorithm. The Ensemble, in addition to Random Forest, is also similar to MLP using the LBFGS algorithm and SVM. MLP using the LBFGS algorithm, in addition to Random Forest and Ensemble, is also similar to SVM. SVM, as already mentioned, is similar to MLP using the LBFGS algorithm and Ensemble. And finally, Naive Bayes and KNN are similar. The other algorithms had different results among themselves.

Random Forest (530) was the algorithm that stood out the most in the results of the recall rank, referring to topology 1, presented in Table 5. Posteriorly, with similar results, are Ensemble (506.5), MLP using LBFGS algorithm (482.5), and SVM (479). Continuing, the MLP using the ADAM algorithm (428.5) which did not have similar results to any algorithm. And finally, Naive Bayes (201.4) and KNN (172), which had similar results.

Table 5- Recall

	Rank (1)	Recall (1)	Rank (2)	Recall (2)
Random Forest	530,0	100,00%	535,5	91,38%
SVM	479,0	99,85%	523,5	91,34%
MLP(LBFGS)	482,5	99,86%	449,0	91,03%
Ensemble	506,5	99,93%	483,0	91,17%
MLP(ADAM)	428,5	99,61%	504,0	91,24%
KNN	172,0	98,94%	205,0	89,56%
Naive Bayes	201,5	99,05%	100,0	74,63%

When applying the Friedman test to recalls collected from the execution of the algorithms in topology 2, a p-value of 2.0309E-105 was obtained, thus rejecting hypothesis H0. Thus, through Conover's post-hoc test, presented in Table 2, it can be stated that Random Forest is similar to SVM and MLP using the ADAM algorithm. SVM, in addition to Random Forest, is also similar to MLP using the ADAM algorithm and Ensemble. The MLP using the ADAM algorithm, in addition to SVM and Random Forest, is also similar to Ensemble. Ensemble, in addition to MLP, using the ADAM algorithm and SVM is also similar to MLP using the LBFGS algorithm. The MLP using the LBFGS algorithm, as already mentioned, is similar to the Ensemble. A MLP utilizando o algoritmo LBFGS, como já foi dito é semelhante ao Ensemble. The other algorithms obtained different results from each other.

Random Forest (535.5) was the algorithm that stood out the most in the results of the recall rank, referring to topology 2, presented in Table 5. Posteriorly, there is SVM (523.5), which is similar to Random Forest, MLP using the ADAM algorithm and Ensemble. Then follows the MLP using the ADAM algorithm (504), which is similar to SVM, Random Forest, and Ensemble. Ensemble (483) is similar to MLP using the ADAM algorithm, SVM, and MLP using the LBFGS algorithm. MLP using the LBFGS algorithm (449) is similar to Ensemble. Finally, KNN (205) and Naive Bayes (100).

3.4 F1-SCORE ANALYSIS

Table 6 presents the results of the machine learning algorithms referring to the f1-score and the f1-score rank of the algorithms in topologies 1 and 2.

Table 6- F1-Score

	Rank (1)	F1-Score (1)	Rank (2)	F1-Score (2)
Random Forest	530,0	100,00%	507,5	91,38%
SVM	479,0	99,85%	532,5	91,48%
MLP(LBFGS)	482,5	99,86%	460,0	91,19%
Ensemble	506,5	99,93%	490,0	91,31%
MLP(ADAM)	428,5	99,58%	508,0	91,37%
KNN	172,0	98,94%	202,0	89,61%
Naive Bayes	201,5	99,05%	100,0	69,10%

When applying the Friedman test to the f1-score collected from the execution of the algorithms in topology 1, a p-value of 7.91246E-89 was obtained, thus rejecting the hypothesis H0. Subsequently, through Conover's post-hoc test, shown in Table 1, it can be stated that Random Forest is similar to Ensemble and MLP using the LBFGS algorithm. Ensemble is also similar to MLP using the LBFGS algorithm and SVM. MLP using the LBFGS algorithm, in addition to Random Forest and Ensemble, is also similar to SVM. SVM, as already mentioned, is similar to MLP using the LBFGS algorithm and Ensemble. Finally, Naive Bayes and KNN are similar. The other algorithms had different results among themselves.

Random Forest (530) was the algorithm that stood out the most in the f1-score rank results, referring to topology 1, presented in Table 6. Posteriorly, with similar results, there is the Ensemble (506.5), the MLP using the LBFGS algorithm (482.5), and the SVM (479). With similar results, right after, there is the Ensemble (506.5), the MLP using the LBFGS algorithm (482.5), and the SVM (479). Then we have the MLP using the ADAM algorithm (428.5) which did not have similar results to any algorithm. Finally, Naive Bayes (201.4) and KNN (172), which had similar results.

When applying the Friedman test to the f1-score collected from the execution of the algorithms in topology 2, the p-value of 3.4897E-107 was obtained, therefore, hypothesis H0 is rejected. Subsequently, through Conover's post-hoc test, presented in Table 2, it can be stated that the SVM obtained results similar to MLP using the ADAM algorithm, Random Forest and Ensemble. The MLP using the ADAM algorithm, in addition to SVM, is similar to Random Forest, Ensemble, and MLP using the LBFGS algorithm. Random Forest, in addition to SVM and MLP using the ADAM algorithm, is also similar to Ensemble and MLP using the LBFGS algorithm. The Ensemble algorithm, in addition to Random Forest, SVM, and MLP using the ADAM algorithm, is also similar to MLP using the LBFGS algorithm. The MLP using the

LBFGS algorithm as mentioned is similar to MLP using the ADAM algorithm, Random Forest, and Ensemble. The other algorithms obtained different results from each other.

SVM (532.5) was the algorithm that stood out the most in the f1-score rank results, referring to topology 2, presented in Table 6. Then, with similar results, are the MLP using the ADAM algorithm (508.0), the Random Forest (507.5), Ensemble (490), and the MLP using the LBFGS algorithm (460). Finally, KNN (202) and Naive Bayes (100).

3.5 TRAINING TIME ANALYSIS

Table 7 presents the results of the machine learning algorithms regarding the training time and the training time rank of the algorithms in topologies 1 and 2.

When applying the Friedman test to the training times collected from the execution of the algorithms in topology 1, a p-value of 3.7286E-124 was obtained, thus rejecting hypothesis H0. Consequently, through Conover's post-hoc test, presented in Table 1, it can be stated that only the Naive Bayes and KNN algorithms had similar results.

Table 7- Training Time

	Rank (1)	Training (1)	Rank (2)	Training (2)
Random Forest	300,0	0,054249s	300,0	0,046911s
SVM	400,0	0,685175s	400,0	2,366524s
MLP(LBFGS)	600,0	25,669977s	600,0	20,725135s
Ensemble	700,0	36,311234s	700,0	31,098941s
MLP(ADAM)	500,0	9,834175s	500,0	8,797501s
KNN	143,5	0,007560s	127,5	0,006079s
Naive Bayes	156,5	0,007811s	172,5	0,006451s

The KNN (143.5) and Naive Bayes (156.5) algorithms stood out in the results of the training time rank, referring to topology 1, presented in Table 7. Then there are the Random Forest (300), SVM (400), the MLP using the ADAM algorithm (500), the MLP using the LBFGS algorithm (600), and the Ensemble (700).

When applying the Friedman test to the training times collected from the execution of the algorithms in topology 2, a p-value of 5.9772E-125 was obtained, thus rejecting hypothesis H0. Therefore, through the Conover post-hoc test presented in Table 2, it can be stated that only the Naive Bayes and KNN algorithms had similar results.

The KNN (127.5) and Naive Bayes (172.5) algorithms stood out in the results of the training time rank, referring to topology 2, presented in Table 7. Then there are the Random Forest (300), SVM (400), the MLP using the ADAM algorithm (500), the MLP using the LBFGS algorithm (600), and the Ensemble (700).

3.6 VALIDATION TIME ANALYSIS

Table 8 presents the results of the machine learning algorithms regarding the validation time and the validation time rank of the algorithms in topologies 1 and 2.

Table 8- Validation Time

	Rank (1)	Validation (1)	Rank (2)	Validation (2)
Random Forest	205,0	0,002749s	216,0	0,002430s
SVM	564,0	0,012541s	600,0	0,036349s
MLP(LBFGS)	455,0	0,010440s	436,0	0,008230s
Ensemble	700,0	0,065563s	700,0	0,081206s
MLP(ADAM)	294,5	0,004432s	282,0	0,003589s
KNN	478,5	0,011070s	464,0	0,008400s
Naive Bayes	103,0	0,001419s	102,0	0,001240s

When applying the Friedman test to the validation times collected from the execution of the algorithms in topology 1, a p-value of 9.8949E-119 was obtained, thus rejecting hypothesis H0. Therefore, through Conover's post-hoc test, presented in Table 1, it can be stated that only the KNN and MLP algorithms using the LBFGS algorithm had similar results.

The Naive Bayes (102) obtained the best validation time followed by Random Forest (205), MLP using the ADAM algorithm (294.5) and MLP using the LBFGS algorithm (455), which obtained a result similar to KNN (478, 5), SVM (564) and Ensemble (700).

When applying the Friedman test to the validation times collected from the execution of the algorithms in topology 2, a p-value of 1.0749E-122 was obtained, thus rejecting hypothesis H0. Consequently, through Conover's post-hoc test presented in Table 2, it can be stated that only the KNN and MLP algorithms using the LBFGS algorithm had similar results.

The Naive Bayes (103) obtained the best validation time, followed by Random Forest (216), MLP using the ADAM algorithm (282), MLP using the LBFGS algorithm (436), which obtained a similar result to KNN (464), SVM (600) or Ensemble (700).

4.0 DISCUSSION

Based on the results that were presented, it is clear that there is a difference about the effectiveness of the algorithms using data from the presented SDN network topologies, depending on the characteristics of the data collected, the result of an algorithm may be better than the others about a certain metric. Table 9 presents the best results when applying the hypothesis test.

Therefore, in topology 1, although Random Forest obtained results similar to other algorithms when verifying certain metrics, it was the one that stood out the most when applying the hypothesis test on conventional metrics (Accuracy, Precision, Recall, F1- Score).

Table 9- Best Results

	Topology 1	Topology 2
Accuracy	Random Forest	Random Forest, SVM
Precision	Random Forest, Ensemble e MLP(LBFGS)	MLP(ADAM) e SVM
Recall	Random Forest, Ensemble e MLP(LBFGS)	Random Forest, SVM e MLP(ADAM)
F1-Score	Random Forest, Ensemble e MLP(LBFGS)	SVM e MLP(ADAM)
Training	KNN e Naive Bayes	KNN e Naive Bayes
Validation	Naive Bayes	Naive Bayes

However, when analyzing the best results in topology 2, it appears that Random Forest obtained results similar to SVM when the hypothesis test was applied to accuracy. About precision, the MLP using the ADAM algorithm obtained results similar to the SVM. About recall, the Random Forest, SVM and MLP using the ADAM algorithm obtained similar results. In the f1-score, SVM and MLP using the ADAM algorithm obtained similar results.

Analyzing the result of the hypothesis test applied to the best model training times, KNN and Naive Bayes obtained similar training times in both topologies. Regarding the best model validation times, Naive Bayes obtained better results in both topologies.

Finally, through the statistical analysis carried out in this work, it can be concluded that Random Forest was the algorithm that stood out the most in conventional metrics and that Naive Bayes stood out about training and validation times. However, as Naive Bayes obtained comparatively lower results, mainly in topology 2, about conventional metrics, it is concluded that Random Forest was the algorithm with the best overall result in this work.

5.0 CONCLUSION

The objective of this work was to carry out a comparative study of machine learning algorithms applied to traffic classification in two SDN topologies. The Mininet platform and the Python language were used to build the two SDN topologies, and the D-ITG was used to generate the traffic from the hosts. Through the data collected by tcpdump and processed by the measurement program, a file was generated with the training and validation patterns that were used by the machine learning algorithms. The performance of the classifiers was evaluated using the conventional metrics of accuracy, precision, recall, and f1-score, in addition to training and validation times. Statistical analysis was carried out by applying the Friedman test and Conover's post-hoc test on the result of conventional metrics and training and validation times. It can be concluded that Random Forest was the algorithm that stood out the most in terms of conventional metrics and that Naive Bayes stood out in terms of training and validation times. However, as Naive Bayes obtained comparatively inferior results, mainly in topology 2, about conventional metrics, it is concluded that Random Forest was the algorithm with the best overall result in this work.

As future works, it is intended to perform a quality of service analysis on the traffic of SDN topologies using the classification data. It is also intended to add new network topologies aiming at new comparisons in the classifiers. Finally, perform a case study to verify the effectiveness of better load balancing based on data generated by machine learning algorithms.

REFERENCES

Amaral, P., Dinis, J., Pinto, P., Bernardo, L., Tavares, J., & Mamede, H. (12 de 2016). Machine learning in software defined networks: Data collection and traffic classification. *IEEE International Conference on Network Protocols*.

Badhani, S., & Muttoo, S. (2019). Cendroid—a cluster-ensemble classifier for detecting malicious android applications. *Computers Security, 85*, pp. 25–40.

Bakker, J., Ng, B., Seah, W., & Pekar, A. (2019). Traffic classification with machine learning in a live network. *IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pp. 488–493.

Bisol, R., Silva, A., Machado, C., Granville, L., & Schaeffer-Filho, A. (2016). Coleta e análise de características de fluxo para classificação de tráfego em redes definidas por software. *XXXIV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*.

Boser, B., Guyon, I., & Vapnik, V. (07 de 1992). A training algorithm for optimal margin classifiers. *Annual Workshop on Computational Learning Theory (COLT'92)* , pp. 144–152.

Botta, A., Dainotti, A., & Pescapé, A. (2012). A tool for the generation of realistic network workload for emerging networking scenarios. *Computer Networks, 56*(15), pp. 3531–3547.

Bužic, D., & Dobša, J. (2018). Lyrics classification using naive bayes. *International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 1011–1015.

Cardoso, W. S., Silva, F., Rocha, U., & Sousa, M. (2017). Implantação de um patch panel virtual utilizando redes definidas por software. *Anais Estendidos do XXIII Simpósio Brasileiro de Sistemas Multimídia e Web*, pp. 95–98.

Chomboon, K., Chujai, P., Teerarassammee, P., Kerdprasop, K., & Kerdprasop, N. (01 de 2015). An empirical study of distance metrics for k-nearest neighbor algorithm. *International Conference on Industrial Application Engineering*, pp. 280–285.

Commiters, A. (04 de 2021). Motor de análise unificado ultrarrápido.

Ding, L., Yu, F., Peng, S., & Xu, C. (4 de 2013). A classification algorithm for network traffic based on improved support vector machine. *Journal of Computers, 8*.

Edla, D., Mangalorekar, K., Dhavalikar, G., & Dodia, S. (2018). Classification of eeg data for human mental state analysis using random forest classifier. *International Conference on Computational Intelligence and Data Science, 132*, pp. 1523–1532.

Fan, Z., & Liu, R. (2017). Investigation of machine learning based network traffic classification. *International Symposium on Wireless Communication Systems (ISWCS)*, pp. 1–6.

Farhady, H., Lee, H., & Nakao, A. (2015). Software-defined networking: A survey. *Computer Networks, 81*, pp. 79–95.

Ferreira, F. R. (2004). *O uso de rede neural artificial mlp na predição de estruturas secundárias de proteínas*. Master's thesis, Universidade Estadual Paulista, Instituto de Biociências, Letras e Ciências Exatas, São José do Rio Preto.

Firmino, M. (2015). *Testes de hipóteses: uma abordagem não paramétrica*. Master's thesis, Universidade de Lisboa, Faculdade de Ciências, Departamento de Estatística e Investigação Operacional.

Gandhi, I., & Pandey, M. (2015). Hybrid ensemble of classifiers using voting. *International Conference on Green Computing and Internet of Things (ICGCIoT)*, pp. 399–404.

García-Pedrajas, N., Romero del Castillo, J., & Cerruela-García, G. (2017). A proposal for local k values for k -nearest neighbor rule. *IEEE Transactions on Neural Networks and Learning Systems, 28*(2), pp. 470–475.

Haykin, S. (2001). *Redes Neurais: princípios e práticas.*

Jedari, E., Wu, Z., Rashidzadeh, R., & Saif, M. (2015). Wi-fi based indoor location positioning employing random forest classifier. *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–5.

Kim, H., & Feamster, N. (02 de 2013). Improving network management with software defined networking. *IEEE Communications Magazine, 51*, pp. 114–119.

Kokila, R., Selvi, S. T., & Govindarajan, K. (2014). Ddos detection and analysis in sdn-based environment using support vector machine classifier. *Sixth International Conference on Advanced Computing (ICoAC)*, pp. 205–210.

Maia, N. (2006). *Engenharia de Tráfego em Domínio MPLs utilizando Técnicas de Inteligência Computacional*. PhD thesis, Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brasil.

Mininet Team. (03 de 2021). *Mininet: An instant virtual network on your laptop (or other pc) - mininet.*

Pandas Development Team. (03 de 2021). *pandas - python data analysis library*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, pp. 2825–283.

Saqlain, M., Jargalsaikhan, B., & Lee, J. (2019). A voting ensemble classifier for wafer map defect patterns identification in semiconductor manufacturing. *IEEE Transactions on Semiconductor Manufacturing, 32*(2), pp. 171–182.

SDN Hub Team. (03 de 2021). *All-in-one sdn app development starter vm | sdn hub.*

Singh, A., Halgamuge, M., & Lakshmiganthan, R. (2017). Impact of different data types on classifier performance of random forest, naïve bayes, and k-nearest neighbors algorithms. *International Journal of Advanced Computer Science and Applications, 8*(12).

Vaidya, J., Shafiq, B., Basu, A., & Hong, Y. (2013). Differentially private naive bayes classification. *IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT),, 1*, pp. 571–576.

Vapnik, V. (1995). *The Nature of Statistical Learning Theory.*

Vapnik, V. (1998). *Statistical Learning Theory.*

Vapnik, V., & Chervonenkis, A. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications, 16*, pp. 264–280.

Vieira, Z., & Bauchspiess, A. (03 de 2016). Implementação do servocontrole autosintonizado em tempo-real utilizando rede perceptron multicamadas. pp. 308–313.

Wu, Z., Xu, Q., Li, J., Fu, C., Xuan, Q., & Xiang, Y. (2018). Passive indoor localization based on csi and naive bayes classification. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 48*, pp. 1566–1577.

3 Considerações Finais

Este trabalho teve como objetivo apresentar estudos de classificação de tráfego em redes definidas por software. Utilizou-se a plataforma Mininet e a linguagem Python para construção das topologias SDN, sendo que o D-ITG foi utilizado para a geração do tráfego dos hosts. Foi construído um programa na linguagem Python para medição do tráfego das aplicações entrantes na topologia SDN. Através dos dados coletados pelo tcpdump e processados pelo programa de medição (promed) foi construído um arquivo com os padrões de treinamento e validação que foram utilizados pelos algoritmos de machine learning. O desempenho dos classificadores foi avaliado principalmente através das métricas convencionais, ou seja, acurácia, precisão, revocação e f1-score. Posteriormente foram adicionados como métrica os tempos de treino e validação do modelo, além de uma análise estatística através da aplicação do teste de Friedman e do teste post-hoc de Conover sobre o resultado das métricas convencionais e dos tempos de treinamento e validação. Pode-se concluir que o Random Forest foi o algoritmo com melhor resultado geral neste trabalho. Vale ressaltar que os resultados apresentados aqui são válidos para as topologias de rede utilizadas nos trabalhos e que em outras topologias os resultados poderiam ser diferentes dos apresentados nesses trabalhos.

Como trabalhos futuros pretende-se fazer uma análise de qualidade de serviço sobre o tráfego das topologias SDN utilizando os dados da classificação. Pretende-se também adicionar novas topologias de rede visando novas comparações nos classificadores. Finalmente, realizar um estudo de caso para verificar a eficácia do melhor balanceamento de carga com base nos dados gerados pelos algoritmos de machine learning.