PEER REVIEW, Vol. 6, Nº 8, 2024
DOI: 10.53660/PRW-2089-3821
ISSN: 1541-1389

Um estudo comparativo de algoritmos de aprendizado de máquina para classificação de tráfego em redes definidas por software

A comparative study of Machine Learning Algorithms for traffic classification in Software Defined Networks

Nilton Alves Maia
ORCID: https://orcid.org/0000-0002-2023-2453
Universidade Estadual de Montes Claros – Unimontes, Brasil
E-mail: nilton.maia@unimontes.br
Victor de Freitas Arruda
ORCID: https://orcid.org/0009-0009-2736-4967
Universidade Estadual de Montes Claros – Unimontes, Brasil
Maurílio José Inácio
ORCID: https://orcid.org/0000-0003-0744-0845
Universidade Estadual de Montes Claros – Unimontes, Brasil
E-mail: maurilio.inacio@unimontes.br
Marilee Patta
ORCID: https://orcid.org/0000-0001-9286-3329
Universidade Estadual de Montes Claros – Unimontes, Brasil
Marcel Veloso Campos
ORCID: https://orcid.org/0000-0002-0581-7314
Universidade Estadual de Montes Claros – Unimontes, Brasil
Narciso Ferreira dos Santos Neto
ORCID: https://orcid.org/0000-0002-1742-3515
Universidade Estadual de Montes Claros – Unimontes, Brasil

RESUMO

As Redes Definidas por Software pode viabilizar o desenvolvimento de técnicas para melhorar o desempenho das redes IP com relação à segurança, qualidade de serviço e engenharia de tráfego. Este trabalho realizou um estudo comparativo de classificação de tráfego em duas topologias SDN utilizando redes neurais artificiais do tipo Multilayer Perceptron, Máquinas de Vetores de Suporte, Naive Bayes, K-Nearest Neighbor, Random Forest e o Ensemble. O desempenho dos classificadores foi avaliado através das métricas acurácia, precisão, revocação e f1-score. Foi realizada uma análise estatística através da aplicação do teste de Friedman e do teste post-hoc de Conover sobre os resultados obtidos pelos algorítmos classificadores. O Random Forest obteve o melhor resultado.

Palavras-chave: Aprendizado de máquina; Redes Definidas por Software; Classificação de Tráfego.

ABSTRACT

Software Defined Networks can enable the development of techniques to improve the performance of IP networks with regard to security, quality of service and traffic engineering. This work carried out a comparative study of traffic classification in two SDN topologies using artificial neural networks of the Multilayer Perceptron type, Support Vector Machines, Naive Bayes, K-Nearest Neighbor, Random Forest and Ensemble. The performance of the classifiers was evaluated using the metrics accuracy, precision, recall and f1-score. A statistical analysis was carried out by applying the Friedman test and the Conover post-hoc test on the results obtained by the classifier algorithms. Random Forest achieved the best result.

Keywords: Machine Learning; Software Defined Networks; Traffic Classification.

Recebido: 05/03/2024 | Aceito: 07/04/2024 | Publicado: 10/04/2024

INTRODUÇÃO

Com o surgimento das redes de computadores houve uma revolução no acesso à informação, fazendo com que se tornassem muito importantes no dia a dia. Elas são utilizadas como suporte nas mais variadas atividades sendo que se transformaram em uma ferramenta vital para o transporte de informações no mundo atual.

Dessa forma é imprescindível que as redes funcionem de forma adequada, uma vez que estão sujeitas a diversos fatores, como sobrecargas, interrupção de serviços, ataques maliciosos, entre outros. Além disso, o desenvolvimento de novas tecnologias nas redes IP tradicionais é muito difícil. Visto que são constituídas por diversos dispositivos, geralmente compostos por software proprietário de determinado fabricante, executando em hardware proprietário, o que faz com que a criação e o desenvolvimento de novas tecnologias para as redes atuais sejam mínimas, tornando-as engessadas (Cardoso, Silva, Rocha, & Sousa, 2017).

Atualmente tem-se observado um aumento da quantidade de aplicações que consomem tráfego, o que acaba produzindo escassez de recursos de rede. Isto faz com que seja necessária a busca pelo desenvolvimento de novas tecnologias que melhorem o desempenho da rede. Desta forma, a identificação dos fluxos de tráfego entrante é de grande importância para o gerenciamento, controle de tráfego e detecção de anomalias na rede (Ding, Yu, Peng, & Xu, 2013).

Neste cenário, as Redes Definidas por Software(SDN - Software Defined Networks) podem facilitar o desenvolvimento de técnicas para melhorar a qualidade de serviço (QoS) da rede. A SDN é caracterizada 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 (Bisol, Silva, Machado, Granville, & Schaeffer-Filho, 2016).

As Redes SDN são um novo paradigma em telecomunicações e redes de computadores. Elas têm o objetivo de auxiliar no enfrentamento de problemas de gerenciamento nas redes IP atuais. Como os administradores de rede são responsáveis pela configuração e aplicação de políticas de alto nível a uma ampla gama de eventos que podem ocorrer, as redes SDN dão esperança da utilização de métodos mais convenientes para a configuração e gerenciamento. A arquitetura SDN é dividida em

três (3) camadas, sendo que no nível mais baixo está o plano de dados; no nível intermediário está o plano de controle e no nível mais alto o plano de aplicações. O plano de dados atua simplesmente como hardware de encaminhamento de pacotes, ele se comunica com o plano de controle através da interface southbound. Essa interface possibilita a comunicação entre os comutadores programáveis e o controlador, ou seja, o controlador usa a interface southbound dos dispositivos comutadores habilitados para SDN para conectar-se ao plano de dados. O plano de controle atua como "cérebro" da rede. Ele é facilmente programável e fornece uma abstração da infraestrutura de rede subjacente. Isto possibilita que os comutadores se tornem dispositivos mais simples, já que eles aceitam as instruções do controlador centralizado. Dessa forma o administrador de rede não precisa configurar os dispositivos de rede individualmente e as decisões de roteamento e encaminhamento são implementadas através do controlador SDN centralizado (Kokila, Selvi, & Govindarajan, 2014). A comunicação entre as aplicações de rede e os controladores é mantida pela interface northbound, que fica localizada no plano de controle. A interface northbound determina como expressar tarefas operacionais e políticas de rede, e também como convertê-las em um formato que o controlador possa entender (Farhady, Lee, & Nakao, 2015). A separação dos planos de dados e controle, propiciada pelas redes SDN, torna possível o desenvolvimento de aplicações visando a melhoria do gerenciamento da rede. Dessa forma, pode-se desenvolver, por exemplo, aplicações para resolver problemas de segurança, qualidade de service (QoS, sigla em inglês) e engenharia de tráfego. Estes problemas podem ser mais bem encaminhados se for possível a obtenção de informações mais precisas sobre o tráfego entrante na rede, informações estas que podem ser obtidas a partir da sua classificação.

A Classificação de Tráfego descreve o processo de identificação e o emparelhamento de fluxos de pacotes para algum tipo de tráfego (tráfego malicioso, tráfego de baixa prioridade, aplicações de rede, etc.). Esse processo depende de informações extraídas do tráfego que serve como entrada para o algoritmo de classificação de tráfego (Bakker, Ng, Seah, & Pekar, 2019). O objetivo da classificação de tráfego é melhorar o gerenciamento de recursos de rede, segurança de rede e QoS. A classificação de tráfego precisa, feita em tempo conveniente está se tornando cada vez mais importante para muitas aplicações em rede com ou sem fio, como por exemplo, engenharia de tráfego, monitoramento de segurança e qualidade de serviço (Fan & Liu,

2017). A classificação de tráfego, pode ser efetuada utilizando números de porta, payload e técnicas de Aprendizado de Máquina.

A classificação por número de porta depende apenas de aplicativos de mapeamento para identificar os números de porta conhecidos. Infelizmente, com o passar do tempo as técnicas de classificação baseadas em números de portas se tornaram imprecisas e algumas limitações se tornaram obvias. Surgiu um alto número de aplicações que não possuem números de porta registrados e muitos deles utilizavam mecanismos dinâmicos de negociação de portas para se esconder de firewalls e ferramentas de segurança de rede (Fan & Liu, 2017; Amaral, 2016).

A classificação baseada em inspeção de payload, também conhecida por Deep Packet Inspection (DPI), é atualmente uma das técnicas mais utilizadas (Amaral, 2016). Os sistemas de DPI utilizam de assinaturas predefinidas de pacotes ou fluxos para descobrir a qual tipo de tráfego o pacote ou fluxo pertence. Dessa forma os sistemas de DPI inspecionam o payload dos pacotes de determinado fluxo para corresponder o pacote ou fluxo com as assinaturas predefinidas, sendo que o processo de correspondência sempre é feito por expressões regulares. Como os métodos baseados em payload necessitam do exame do payload de cada um dos pacotes, as vezes ocorrem alguns problemas, como as leis de privacidade e a criptografia que podem levar a um payload de tráfego inacessível. Por outro lado, a inspeção profunda de pacotes (DPI) resulta em altos custos computacionais e requer manutenção manual de assinaturas (Fan & Liu, 2017).

Os métodos de classificação baseados em Aprendizado de Máquina podem superar algumas das limitações de abordagens baseadas em porta e payload. Mais especificamente, as técnicas de Aprendizado de Máquina podem classificar o tráfego da Internet usando estatísticas independentes do protocolo da aplicação, como duração do fluxo, variação do comprimento do pacote, tamanho máximo ou mínimo do segmento, tamanho da janela, tempo de ida e volta e tempo de chegada de pacotes. Além disso, pode levar a custos computacionais mais baixos e identificar o tráfego criptografado facilmente (Fan & Liu, 2017). A classificação do tráfego pode ser realizada usando aprendizado supervisionado ou não supervisionado. No aprendizado supervisionado, é necessário obter conjuntos de dados de treinamento rotulados em que novas aplicações podem aparecer. Já o aprendizado de máquina não supervisionado é normalmente utilizado para tarefas de clustering, onde os algoritmos agrupam os dados em diferentes

clusters de acordo com as semelhanças nos valores dos recursos. O objetivo é identificar relacionamentos desconhecidos nos dados, encontrando padrões de similaridade entre as várias observações (Amaral, Dinis, Pinto, Bernardo, Tavares, & Mamede, 2016). Os classificadores que foram utilizados neste trabalho utilizam o aprendizado supervisionado. Mais especificamente, são utilizados modelos de rede neural artificial (RNA) do tipo MLP (MultilayerPerceptron) utilizando o ADAM e o LBFGS (Limited-Memory Broyden-Fletcher-Goldfarb-Shanno), o SVM (Support Vector Machine), Naive Bayes, KNN(K-Nearest Neighbor), Random Forest e o Ensemble(Votação).

As RNAs do tipo MLP fazem com que seja possível a resolução de problemas complexos, os quais não são possíveis de serem resolvidos pelo modelo de neurônio básico. Os neurônios internos da MLP são muito importantes, pois é comprovado que sem estes é impossível a resolução de problemas não linearmente separáveis (Ferreira, 2004). A estrutura da rede MLP é formada pelas entradas, uma ou mais camadas intermediárias e a camada de saida. As saídas dos neurônios de cada camada são utilizadas como entrada para a camada posterior.

A classificação através das Máquinas de Vetores de Suporte (SVM - Support Vector Machine) consiste na separação ótima de um grupo de dados, independentemente da sua dimensionalidade, através de um problema de programação quadrática que permite boa generalização (Vapnik, 1998). Esse processo faz com que a SVM encontre um mínimo global na superfície de custo, sendo considerado como uma vantagem do método.

O Naive Bayes é um algoritmo de Aprendizado de Máquina, onde o classificador aprende por meio de um algoritmo de classificação de documentos (Bužic & Dobša, 2018). O classificador Naive Bayes se baseia em duas suposições básicas: 1) as características são independentes umas das outras e 2) cada característica tem a mesma proeminência (Wu, Xu, Li, Fu, Xuan, & Xiang, 2018). O classificador Naive Bayes utiliza um número arbitrário de variáveis contínuas ou categóricas e classifica uma instância para pertencer a uma das várias classes. Assim, ele se aplica a tarefas de aprendizagem onde cada instância x é descrita por uma conjunção de valores de atributos e a função alvo f(x) (Vaidya, Shafiq, Basu, & Hong, 2013). Este classificador é baseado no teorema de Bayes. A eficiência na modelagem e previsão é uma vantagem inquestionável sobre outros algoritmos de classificação, que se deve à possibilidade de

fácil paralelização, especialmente importante para grandes conjuntos de dados (Bužic & Dobša, 2018).

O K-Nearest Neighbor (KNN) é um algoritmo que utiliza agrupamento para efetuar a classificação dos dados, dessa forma ele pode utilizar de aprendizado supervisionado ou não supervisionado para efetuar a classificação, no contexto deste trabalho foi utilizado o aprendizado supervisionado de modo que ele requer dados de treinamento e um valor k predefinido para encontrar os k dados mais próximos com base no cálculo da distância (Chomboon, Chujai, Teerarassammee, Kerdprasop, & Kerdprasop, 2015). O método KNN, embora simples, geralmente pode corresponder e até superar métodos mais sofisticados e complexos em termos de erro de generalização. Um dos problemas com este classificador, entretanto, é fixar o valor apropriado de k (García-Pedrajas, Romero del Castillo, & Cerruela-García, 2017). O KNN constrói previsões diretamente do conjunto de dados de treinamento, que é armazenado na memória. Para classificar um dado desconhecido, por exemplo, KNN encontra o conjunto de k objetos dos dados de treinamento mais próximos da instância de dados de entrada por um cálculo de distância e atribui o máximo de classes votadas dessas classes vizinhas (Singh, Halgamuge, & Lakshmiganthan, 2017).

O classificador Random Forest usa várias árvores de decisão durante a fase de treinamento e produz a previsão média de árvores individuais (Edla, Mangalorekar, Dhavalikar, & Dodia, 2018). Para prever o valor alvo para uma nova instância de dados, a nova observação é alimentada para todas as árvores de classificação na Random Forest. Os números de predição para uma classe realizada por cada uma das árvores de classificação são contados. Então, a classe com o número máximo de votos é retornada como o rótulo da classe para a nova instância de dados vizinhas (Singh, Halgamuge, & Lakshmiganthan, 2017). Existem algumas propriedades que tornam Random Forest um modelo de classificação muito bom para grandes conjuntos de dados, como (a) sem necessidade de poda de árvores, (b) geração automática de precisão e importância variável, (c) não sendo muito sensível a outliers nos dados de treinamento, e (d) ser um modelo fácil de definir parâmetros (Jedari, Wu, Rashidzadeh, & Saif, 2015).

Os métodos Ensemble combinam previsões de vários classificadores básicos e fornecem a previsão final. O modelo ensemble combina o conjunto de classificadores para criar um único modelo composto que fornece melhor acurácia. Os métodos Ensemble podem ser definidos como comitê, fusão de classificador, combinação ou

agregação, votação, etc. Pesquisas mostram que a previsão de um modelo composto fornece melhores resultados em comparação com a previsão de um único modelo (Gandhi & Pandey, 2015). O desempenho dos métodos ensemble depende da acurácia dos classificadores individuais e do número de classificadores de base incluídos (ou seja, quanto mais classificadores incluímos, melhor será o desempenho do classificador ensemble) (Saqlain, Jargalsaikhan, & Lee, 2019). Os métodos Ensemble podem ser classificados como Métodos Ensemble homogêneos e métodos Ensemble heterogêneos. Os métodos Ensemble homogêneos usam um único algoritmo de aprendizagem em conjuntos de dados de treinamento diferentes para construir vários classificadores, como Bagging, Boosting, Random Subspacesand Random Forest, etc. Enquanto métodos Ensemble heterogêneos usam vários algoritmos de aprendizagem de máquina e manipulam conjuntos de dados de treinamento para fazer vários modelos. Alguns dos métodos heterogêneos são votação, stacking, etc. (Gandhi & Pandey, 2015). Neste trabalho se utilizou o método Ensemble heterogêneo de votação. No método de votação, a previsão de cada classificador base é contada como um voto para uma classe, e a classe final atribuída é aquela que arrecada a maioria dos votos (Badhani & Muttoo, 2019).

Com o intuito de verificar se existe diferença significativa entre os resultados dos classificadores empregados aplicou-se o teste de Friedman e o teste post-hoc de Conover. O teste de Friedman é um teste não paramétrico, ou seja, ele não assume uma distribuição a partir dos indicadores de desempenho, logo este teste é utilizado para verificar se os dados analisados são estatisticamente similares. Assim foram propostas a hipótese nula (H0), que assume que as distribuições das k amostras são idênticas, e a Hipótese Alternativa (H1), que afirma que as distribuições das k amostras diferem na localização (Firmino, 2015). Caso a hipótese nula seja rejeitada, é possível a utilização de testes post-hoc para verificar quais dos grupos analisados são diferentes entre si. Neste caso foi utilizado o teste post-hoc de Conover, que é capaz de encontrar diferenças estatisticamente significativas entre os resultados dos classificadores.

Este trabalho propõe a implementação de um sistema de classificação do tráfego de aplicações entrantes em duas topologias SDN. As aplicações são Voip, Telnet, Quake3, DNS, CSi, CSa, Vídeo e Web. Para executar a classificação foram comparados os modelos de RNA do tipo MLP (utilizando o ADAM e o LBFGS), o SVM, o KNN, o Naive Bayes, Random Forest e o Ensemble.

As próximas seções deste artigo são organizadas da seguinte maneira: na próxima seção é apresentada a metodologia. Em seguida, são apresentados os resultados obtidos com os experimentos. Finalmente, são apresentadas as conclusões sobre o trabalho.

METODOLOGIA

Inicialmente faz-se a medição do tráfego das aplicações entrantes na topologia SDN. Os dados resultantes da medição são armazenados em um arquivo. Estes dados que são utilizados pelos algoritmos de Aprendizado de Máquina para a realização das classificações. Um resumo da metodologia utilizada neste trabalho é mostrado na Figura 1.

Figura 1- Metodologia utilizada neste trabalho

A simulação das topologias SDN foi realizada com a utilização do software Mininet na plataforma SDN HUB. O código para implementação das topologias SDN no Mininet foi escrito em linguagem Python. O software D-ITG (Distribuited Internet Traffic Generator) (Botta, Dainotti, & Pescapé, 2012) foi utilizado para gerar o tráfego das aplicações a partir dos hosts. Para a captura do tráfego foi utilizado o tcpdump. Ele é baseado na libpcap, uma API para a captura de pacotes de rede. Com a captura do tráfego o tcpdump gera um arquivo no formato pcap, este arquivo é transformado em csv pelo tshark, que é uma implementação do Wireshark em modo texto, é muito similar ao tcpdump, porém permite a conversão do arquivo pcap para csv.

As etapas utilizadas para a execução da medição do tráfego são descritas na figura 2. Primeiramente é iniciado o controlador floodlight. Em seguida, a topologia SDN é iniciada no software Mininet. A aplicação tcpdump começa a capturar o tráfego que trafega pelas interfaces de rede. Com a topologia SDN em funcionamento o D-ITG começa a gerar o tráfego na rede. Como o tcpdump não converte o arquivo pcap em csv, utiliza-se o tshark para converter o arquivo para o tipo csv. O processador de medidas (Promed) processa o arquivo csv e gera a medição de tráfego.

Figura 2- Etapas para execução da medição de tráfego

Nas topologias de rede SDN adotadas neste trabalho, os hosts foram configurados como receptores e geradores de tráfego. Cada um dos hosts geradores transmite pacotes de tráfego de uma aplicação de rede diferente. As aplicações de rede selecionadas foram Voip, Telnet, Quake3, DNS, CSa (CounterStrike Ativo), CSi (CounterStrike Inativo), Web, e Vídeo. Voip é a aplicação de voz sobre IP. Telnet é uma aplicação que permite acesso remoto a qualquer máquina que esteja rodando o módulo servidor. Quake3 é um jogo eletrônico de tiro em primeira pessoa. O DNS simula uma aplicação que é responsável por localizar e traduzir para números IP os endereços dos sites digitados nos navegadores. CSa e CSi são jogos de tiro em primeira pessoa que simulam quando o usuário está ativo e inativo, respectivamente. Web e Vídeo simulam respectivamente aplicações web e vídeo.

No promed foi utilizada a biblioteca pandas e o spark para o processamento de grande quantidade de dados. A medição gerada foi feita através da aplicação de métricas nas informações presentes nos cabeçalhos dos pacotes, sendo que ao final do processamento foi gerado um novo arquivo e esse arquivo de medição é utilizado como entrada pelos algoritmos de classificação.

A topologia SDN ficou em funcionamento durante 1 hora, com isso cada medição do tráfego teve duração de 10 segundos, sendo que no final foram obtidos trezentas e sessenta (360) padrões de medições. Os trezentos e sessenta padrões de medições de cada aplicação foram separados em dois conjuntos de dados, sendo utilizados setenta e cinco por cento(75%) para o treinamento dos algoritmos e os outros vinte e cinco por cento(25%) para validação.

Os atributos dos padrões de medição utilizados pelos algoritmos de Machine Learning durante o treinamento e a validação foram o atraso(delay) médio, variação do

atraso(jitter) médio, vazão(throughput), taxa de pacotes média por segundo, quantidade de pacotes transmitidos e a quantidade de bytes transmitidos. As saídas dos algoritmos são as aplicações de rede (classes de medição de tráfego) identificadas. Os classificadores foram selecionados com base em outros trabalhos existentes na literatura. Os algoritmos das RNAs do tipo MLP (ADAM e LBFGS), SVM, KNN, Naive Bayes, Random Forest e Ensemble foram implementados utilizando a linguagem Python e a biblioteca Scikit-learn (Pedregosa, et al., 2011).

A rede MLP com o ADAM foi configurada com três(3) camadas escondidas com cem(100), cinquenta(50) e cinquenta(50) neurônios, respectivamente. Além disso, o modelo utilizou a função de ativação tangente hiperbólica(tanh) e o parâmetro de penalidade(alpha) igual a 0.0001. Já a rede MLP com o LBFGS foi configurada com duas (2) camadas escondidas com cem(100) neurônios cada uma. O modelo utilizou a função de ativação logística (logistic) e o parâmetro de penalidade(alpha) com valor 0.05. Nos dois modelos foi utilizada o tipo de taxa de aprendizado invscaling e a quantidade máxima de interações igual a quinhentos (500).

Para o algoritmo SVM a complexidade(C) foi definida como mil(1000), a relevância dos dados mais próximos a fronteira de separação(gamma) foi definida como sendo um(1) e o kernel utilizado foi o RBF (Radial Basis Function).

No Random Forest foram usadas dez (10) árvores na floresta (n_estimators). O número mínimo de amostras necessárias para dividir um nó interno(min_samples_split) foi definida como dez(10). A profundidade máxima da árvore (max_depth) foi definida como nulo e o número de características considerados ao procurar a melhor divisão(max_features) foi a raiz quadrada do número de características.

No KNN, a quantidade de vizinhos selecionada(n_neighbors) foi definida como três(3) e para a distância(p) foi selecionada Manhattan. O algoritmo utilizado para computar o vizinho mais próximo(algorithm) foi definido como "auto" e a função de pesos(weights) definida na predição foi o inverso da distância.

No Naive Bayes, o valor da maior variação de todos os recursos que é adicionada às variações para estabilidade do cálculo (var_smoothing) foi definido como 1.519911e-06.

No caso do Ensemble utilizou-se o Ensemble de votação com os melhores parâmetros encontrados pela validação cruzada.

RESULTADOS

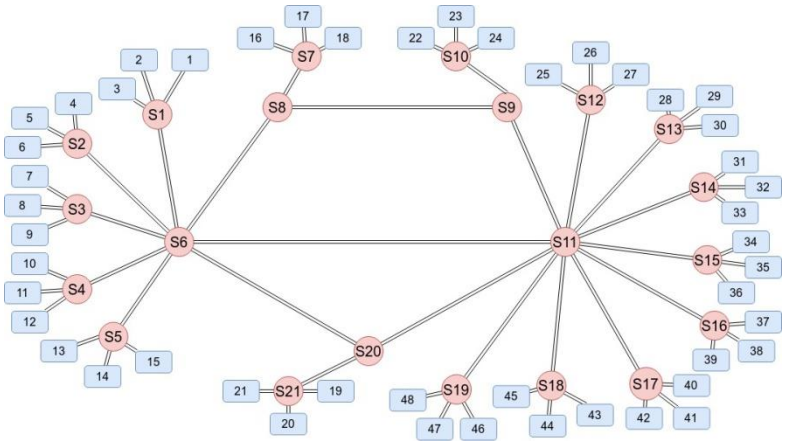
Nesta seção são apresentados os resultados da avaliação dos algoritmos de classificação do trafego entrante na topologia SDN. Foram utilizadas duas topologias de redes diferentes adaptadas a partir do trabalho de (Maia, 2006).

A primeira topologia de rede utilizada é apresentada na figura 3. Ela é formada por quarenta e oito (48) hosts, vinte e um (21) comutadores (switches) e um controlador. Os hosts foram divididos entre clientes e servidores (receptores), sendo vinte e quatro (24) clientes e vinte e quatro (24) servidores. A largura de banda dos enlaces entre os hosts e os comutadores foi configurada em 100 Mbps.

A segunda topologia de rede utilizada é apresentada na figura 4. Ela é formada por cinquenta (50) hosts, dezesseis (16) comutadores (switches) e um controlador. Os hosts foram divididos entre clientes e servidores (receptores), sendo vinte e cinco (25) clientes e vinte e cinco (25) servidores. A largura de banda dos enlaces entre os hosts e os comutadores foi configurada em 100Mbps.

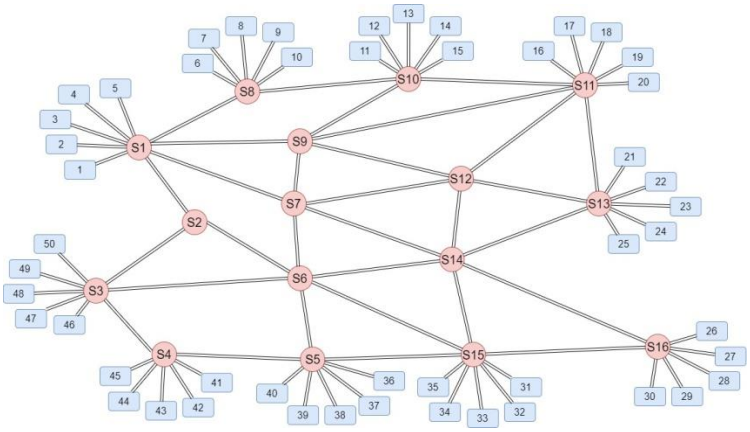
Para efetuar a análise dos resultados utilizou-se de teste de hipótese, sendo assim propostas duas hipóteses. A hipótese nula (H0) diz que as medianas dos resultados dos algoritmos utilizando-se a métrica analisada são todas iguais, ou seja, os resultados de todos os algoritmos de Aprendizado de Máquina, analisando-se a métrica em questão, são estatisticamente os mesmos. Já a hipótese H1 diz nem todas as medianas dos resultados dos algoritmos utilizando-se a métrica analisada são iguais, ou seja, ao menos um resultado de um algoritmo de Aprendizado de Máquina, analisando-se a métrica em questão, difere dos demais.

Figura 3- Topologia 1



Fonte: Adaptado de (Maia, 2006)

Figura 4- Topologia 2



Fonte: Adaptado de (Maia, 2006)

Propostas as hipóteses, aplicou-se o teste de Friedman em todas as métricas selecionadas utilizando 5% como nível de significância e 6 como grau de liberdade. Caso o p-valor encontrado for menor que o nível de significância, rejeita-se a hipótese H0 e se faz o teste post-hoc de Conover para verificar quais são os algoritmos que tem resultados diferentes entre si.

As hipóteses foram feitas sobre as métricas Acurácia, Precisão, Revocação e F1-Score. Cada um dos algoritmos de classificação foi executado 100 (cem) vezes e os algoritmos utilizaram os mesmos parâmetros ao processar os dados de entrada das duas topologias. Para cada execução dos algoritmos os dados de entrada foram separados em dados de treinamento e validação, sendo que para efetuar a análise estatística cada um dos algoritmos teve que utilizar o mesmo conjunto de dados que os demais em cada

execução. As Tabelas 1 e 2 apresentam os resultados do p-valor do Teste de Conover sobre as métricas na Topologia 1 e 2, respectivamente. Caso o p-valor encontrado for menor que o nível de significância dividido pela quantidade de comparações (0.05/21=0.00238), diz-se que os dois algoritmos comparados tiveram resultados diferentes, caso contrário eles obtiveram resultados semelhantes.

Tabela 1- Resultados do p-valor do Teste de Conover sobre as métricas na Topologia 1

Comparação	Acurácia	Precisão	Revocação	F1-Score
adam –lbfgs	1,43905E-08	5,34054E-06	8,72234E-04	3,63345E-03
adam –svm	5,63385E-04	5,81806E-03	2,36003E-01	1,36650E-01
adam – randomforest	3,59177E-04	1,52419E-25	5,58156E-02	9,75745E-01
adam - Naive Bayes	2,17318E-92	1,58540E-114	1,49921E-92	7,69300E-94
adam - k-nn	3,50428E-60	3,07783E-82	4,26390E-59	2,85772E-61
adam – ensemble	1,52556E-02	1,78423E-04	2,01930E-01	2,73966E-01
lbfgs – svm	5,27745E-19	6,85062E-02	7,06611E-06	1,22521E-05
lbfgs – randomforest	1,94727E-19	4,10230E-10	1,99263E-07	3,99871E-03
lbfgs - Naive Bayes	3,31429E-62	2,71165E-90	7,34180E-75	2,20488E-78
lbfgs - k-nn	1,16503E-32	2,52814E-58	1,27433E-42	1,07879E-46
lbfgs – ensemble	9,70965E-04	4,11839E-01	3,90431E-02	6,85062E-02
svm – randomforest	9,03205E-01	1,70342E-15	4,65683E-01	1,28838E-01
svm - Naive Bayes	1,05640E-110	5,82610E-100	7,79590E-99	9,91890E-102
svm - k-nn	3,18978E-78	1,04854E-67	3,49234E-65	5,68380E-69
svm – ensemble	6,08152E-09	3,15913E-01	1,40325E-02	9,96404E-03
randomforest - Naive Bayes	2,42940E-111	4,32207E-57	1,07730E-102	1,11507E-93
randomforest - k-nn	7,27979E-79	1,62071E-28	5,68380E-69	4,09008E-61
randomforest - ensemble	3,01211E-09	2,11691E-12	1,47889E-03	2,87502E-01
Naive Bayes - k-nn	1,47411E-09	1,76442E-09	3,40713E-10	1,02665E-09
Naive Bayes - ensemble	1,66007E-79	1,20260E-94	8,84222E-86	4,90199E-88
k-nn – ensemble	5,05914E-48	1,61439E-62	1,15060E-52	1,03915E-55

Tabela 2- Resultados do p-valor do Teste de Conover sobre as métricas na Topologia 2

Comparação	Acurácia	Precisão	Revocação	F1-Score
adam – lbfgs	5,13730E-09	5,42888E-03	6,57408E-04	6,57408E-04
adam – svm	3,67725E-02	1,04516E-02	1,43316E-03	1,43316E-03
adam – randomforest	1,11894E-40	1,05834E-08	2,50474E-10	2,50474E-10
adam - Naive Bayes	1,60145E-32	2,20898E-40	1,57245E-40	1,57245E-40
adam - k-nn	3,31790E-64	1,83165E-52	1,73237E-49	1,73237E-49
adam – ensemble	1,19674E-07	2,47160E-06	9,82089E-07	9,82089E-07
lbfgs – svm	1,37788E-04	8,24402E-01	8,24402E-01	8,24402E-01
lbfgs – randomforest	1,54734E-16	2,69997E-03	2,69997E-03	2,69997E-03
lbfgs - Naive Bayes	6,01671E-61	6,92001E-54	3,09459E-57	3,09459E-57
lbfgs - k-nn	2,06954E-95	1,14942E-66	7,87038E-67	7,87038E-67
lbfgs – ensemble	5,68338E-01	4,97461E-02	1,28495E-01	1,28495E-01
svm – randomforest	2,73988E-31	1,28559E-03	1,28559E-03	1,28559E-03
svm - Naive Bayes	5,15324E-42	8,85889E-53	4,08807E-56	4,08807E-56
svm - k-nn	4,09424E-75	1,62289E-65	1,11225E-65	1,11225E-65
svm – ensemble	1,15218E-03	2,90483E-02	8,16469E-02	8,16469E-02
randomforest - Naive Bayes	1,22720E-105	2,64556E-69	8,75136E-73	8,75136E-73
randomforest - k-nn	1,48630E-139	1,79129E-82	1,21742E-82	1,21742E-82
randomforest - ensemble	1,73060E-18	2,95755E-01	1,36631E-01	1,36631E-01
Naive Bayes - k-nn	1,13989E-10	1,25026E-02	6,18340E-02	6,18340E-02
Naive Bayes - ensemble	4,86848E-58	7,04524E-64	5,03737E-65	5,03737E-65
k-nn – ensemble	2,19341E-92	5,98438E-77	8,81924E-75	8,81924E-75

A seleção dos melhores parâmetros para os algoritmos selecionados nos cenários em questão foi feita através do Scikit-learn (Pedregosa, et al., 2011) utilizando a validação cruzada GridSearch, onde foi realizada uma pesquisa exaustiva sobre valores de parâmetros especificados para cada algoritmo, apresentando os parâmetros que obtiveram melhores resultados. Utilizou-se os dados coletados da topologia 2 para se fazer a validação cruzada, visto que eles estavam obtendo os valores mais baixos referente as métricas selecionadas.

ANÁLISE DE ACURÁCIA

A Tabela 3 apresenta os resultados dos algoritmos de Aprendizado de Máquina referentes a acurácia e ao rank da acurácia dos algoritmos nas topologias 1 e 2.

Tabela 3- Acurácia nas topologias (1) e (2)

	Rank (1)	Acurácia(1)	Rank(2)	Acurácia(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%

Ao aplicar o teste de Friedman nas acurácias coletadas da execução dos algoritmos na topologia 1 se obteve o p-valor de 2,08086E-91, logo rejeita-se a hipótese H0. Com isso, através do teste post-hoc de Conover, apresentados na Tabela 1, pode-se afirmar que o SVM e a RNA do tipo MLP utilizando o algoritmo ADAM tem resultados semelhantes. O Ensemble e a MLP utilizando o algoritmo LBFGS também apresentaram resultados semelhantes. Os demais algoritmos obtiveram resultados diferentes entre si. O Random Forest (636,5) foi quem mais se destacou no rank de acurácia, referente a topologia 1, apresentado na Tabela 3.

Ao aplicar o teste de Friedman nas acurácias coletadas da execução dos algoritmos na topologia 2 se obteve o p-valor de 6,93952E-89, logo rejeita-se a hipótese H0. Com isso, através do teste post-hoc de Conover, apresentados na Tabela 2, pode-se afirmar que o SVM e o Random Forest tem resultados semelhantes. A MLP utilizando o algoritmo ADAM e o Ensemble também apresentaram resultados semelhantes. Os demais algoritmos obtiveram resultados diferentes entre si. O Random Forest (562,5) e o SVM (560,5), com resultados semelhantes entre si, destacaram-se no rank de acurácia, referente a topologia 2, apresentado na Tabela 3.

ANÁLISE DE PRECISÃO

A Tabela 4 apresenta os resultados dos algoritmos de Aprendizado de Máquina referentes a precisão e ao rank da precisão dos algoritmos nas topologias 1 e 2.

Tabela 4- Precisão nas topologias (1) e (2)

	Rank (1)	Precisão (1)	Rank (2)	Precisão (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%

Ao aplicar o teste de Friedman nas precisões coletadas da execução dos algoritmos na topologia 1 se obteve o p-valor de 2,03667E-88, logo rejeita-se a hipótese H0. Com isso, através do teste post-hoc de Conover podemos afirmar que o Random Forest é semelhante ao Ensemble e a MLP utilizando o algoritmo LBFGS. O Ensemble, além do Random Forest, é semelhante a MLP utilizando o algoritmo LBFGS e ao SVM. A MLP utilizando o algoritmo LBFGS, além do Random Forest e Ensemble, é também semelhante a MLP utilizando o algoritmo ADAM e ao SVM. O SVM, além do Ensemble e a MLP utilizando o LBFGS, também é semelhante a MLP utilizando o algoritmo ADAM. A MLP utilizando o algoritmo ADAM, como foi dito é semelhante a MLP utilizando o algoritmo LBFGS e ao SVM. Por fim o Naive Bayes e o KNN são semelhantes entre si, os demais algoritmos tiveram resultados diferentes entre si. O Random Forest (526) foi o destaque nos resultados do rank de precisão, referente a topologia 1, apresentados na Tabela 4.

Ao aplicar o teste de Friedman nas precisões coletadas da execução dos algoritmos na topologia 2 se obteve o p-valor de 6,8256E-104, logo rejeita-se a hipótese H0. Com isso, através do teste post-hoc de Conover, apresentados na Tabela 2, pode-se afirmar que a MLP utilizando o algoritmo ADAM obteve resultados semelhantes ao SVM. O SVM, além da MLP utilizando algoritmo ADAM, é semelhante ao Ensemble e a MLP utilizando o algoritmo LBFGS. O Ensemble, além do SVM, é semelhante a MLP utilizando o LBFGS. A MLP utilizando algoritmo LBFGS, como foi dito tem resultados semelhantes ao SVM e ao Ensemble. Os demais algoritmos obtiveram resultados diferentes entre si. A MLP utilizando o algoritmo ADAM (572,5) foi o destaque nos resultados do rank de precisão, referente a topologia 2, apresentados na Tabela 4.

ANÁLISE DE REVOCAÇÃO

A Tabela 5 apresenta os resultados dos algoritmos de Aprendizado de Máquina referentes a revocação e ao rank da revocação dos algoritmos nas topologias 1 e 2.

Ao aplicar o teste de Friedman nas revocações coletadas da execução dos algoritmos na topologia 1 se obteve o p-valor de 7,91246E-89, logo rejeita-se a hipótese H0. Com isso, através do teste post-hoc de Conover, apresentados na Tabela 1, pode-se afirmar que o Random Forest é semelhante ao Ensemble e a MLP utilizando o algoritmo LBFGS. O Ensemble, além do Random Forest, também é semelhante a MLP utilizando o algoritmo LBFGS e ao SVM. A MLP utilizando o algoritmo LBFGS, além do Random Forest e o Ensemble, também é semelhante ao SVM. O SVM, como já foi dito, é semelhante a MLP utilizando o algoritmo LBFGS e ao Ensemble. E por fim, o Naive Bayes e o KNN são semelhantes. Os demais algoritmos tiveram resultados diferentes entre si. O Random Forest (530) foi o destaque nos resultados do rank de revocação, referente a topologia 1, apresentados na Tabela 5.

Tabela 5- Revocação nas topologias (1) e (2)

	Rank (1)	Revocação (1)	Rank (2)	Revocação (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%

Ao aplicar o teste de Friedman nas revocações coletadas da execução dos algoritmos na topologia 2 se obteve o pvalor de 2,0309E-105, logo rejeita-se a hipótese H0. Com isso, através do teste post-hoc de Conover, apresentados na Tabela 2, pode-se afirmar que o Random Forest é semelhante ao SVM e a MLP utilizando o algoritmo ADAM. O SVM, além do Random Forest, também é semelhante a MLP utilizando o algoritmo ADAM e ao Ensemble. A MLP utilizando o algoritmo ADAM, além do SVM e Random Forest, também é semelhante ao Ensemble. O Ensemble, além da MLP utilizando o algoritmo ADAM e o SVM, também é semelhante a MLP utilizando o algoritmo LBFGS. A MLP utilizando o algoritmo LBFGS, como já foi dito é semelhante ao Ensemble. Os demais algoritmos obtiveram resultados diferentes entre si. O Random Forest (535,5) foi o destaque nos resultados do rank de revocação, referente a topologia 2, apresentados na Tabela 5.

ANÁLISE DE F1-SCORE

A Tabela 6 apresenta os resultados dos algoritmos de Aprendizado de Máquina referentes ao f1-score e ao rank do f1-score dos algoritmos nas topologias 1 e 2.

Tabela 6- F1-Score nas topologias (1) e (2)

	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%

Ao aplicar o teste de Friedman aos f1-score coletados da execução dos algoritmos na topologia 1 se obteve o p-valor de 7,91246E-89, logo rejeita-se a hipótese H0. Com isso, através do teste post-hoc de Conover, apresentados na Tabela 1, pode-se afirmar que o Random Forest é semelhante ao Ensemble e a MLP utilizando o algoritmo LBFGS. O Ensemble também é semelhante a MLP utilizando o algoritmo LBFGS e o SVM. A MLP utilizando o algoritmo LBFGS, além do Random Forest e o Ensemble, também é semelhante ao SVM. O SVM é semelhante a MLP utilizando o algoritmo LBFGS e ao Ensemble. Por fim, o Naive Bayes e o KNN são semelhantes. Os demais algoritmos tiveram resultados diferentes entre si. O Random Forest (530) foi o destaque nos resultados do rank de f1-score, referente a topologia 1, apresentados na Tabela 6.

Ao aplicar o teste de Friedman aos f1-score coletados da execução dos algoritmos na topologia 2 se obteve o p-valor de 3,4897E-107, logo rejeita-se a hipótese H0. Com isso, através do teste post-hoc de Conover, apresentados na Tabela 2, pode-se afirmar que o SVM obteve resultados semelhantes a MLP utilizando o algoritmo ADAM, ao Random Forest e ao Ensemble. A MLP utilizando o algoritmo ADAM, além do SVM, é semelhante ao Random Forest, Ensemble e a MLP utilizando o algoritmo LBFGS. O Random Forest, além do SVM e da MLP utilizando o algoritmo ADAM, também é semelhante ao Ensemble e a MLP utilizando o algoritmo LBFGS. O Ensemble, além do Random Forest, do SVM e da MLP utilizando o algoritmo ADAM, também é semelhante a MLP utilizando o algoritmo LBFGS. A MLP utilizando o algoritmo LBFGS é semelhante a MLP utilizando o algoritmo ADAM, ao Random Forest e ao Ensemble. Os demais algoritmos obtiveram resultados diferentes entre si. O

SVM (532,5) foi o destaque nos resultados do rank de f1-score, referente a topologia 2, apresentados na Tabela 6.

CONCLUSÃO

Este trabalho teve como objetivo realizar um estudo comparativo de algoritmos de Aprendizado de Máquina aplicados a classificação de tráfego em duas topologias SDN. Utilizou-se a plataforma Mininet e a linguagem Python para construção das topologias, sendo que o D-ITG foi utilizado para a geração do tráfego dos hosts. Os dados de tráfego gerados pelas aplicações entrantes nas topologias SDN foram coletados pelo tcpdump e processados pelo programa de medição, e em seguida foi gerado um arquivo com os padrões de treinamento e validação que depois foram utilizados pelos algoritmos de Aprendizado de Máquina. O desempenho dos classificadores foi avaliado através das métricas convencionais acurácia, precisão, revocação e f1-score. Foi realizada uma análise estatística através da aplicação do teste de Friedman e do teste post-hoc de Conover sobre os resultados obtidos pelos algorítmos classificadores nas métricas convencionais. Pode-se concluir que o Random Forest foi o algoritmo que mais se destacou, sendo que obteve o melhor resultado geral neste trabalho.

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 Aprendizado de Máquina.

REFERÊNCIAS

AMARAL, P., DINIS, J., PINTO, P., BERNARDO, L., TAVARES, J., & MAMEDE, H. Machine learning in software defined networks: Data collection and traffic classification. IEEE International Conference on Network Protocols, 2016.

BADHANI, S., & MUTTOO, S. Cendroid—a cluster-ensemble classifier for detecting malicious android applications. Computers Security, 85, pp. 25–40, 2019.

BAKKER, J., NG, B., SEAH, W., & PEKAR, A. Traffic classification with machine learning in a live network. IFIP/IEEE Symposium on Integrated Network and Service Management (IM), pp. 488–493, 2019.

BISOL, R., SILVA, A., MACHADO, C., GRANVILLE, L., & SCHAEFFER-FILHO, A. 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, 2016.

BOSER, B., GUYON, I., & VAPNIK, V. A training algorithm for optimal margin classifiers. Annual Workshop on Computational Learning Theory (COLT'92) , pp. 144–152, 1992.

BOTTA, A., DAINOTTI, A., & PESCAPÉ, A. A tool for the generation of realistic network workload for emerging networking scenarios. Computer Networks, 56(15), pp. 3531–3547, 2012

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

CARDOSO, W. S., SILVA, F., ROCHA, U., & SOUSA, M. 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, 2017.

CHOMBOON, K., CHUJAI, P., TEERARASSAMMEE, P., KERDPRASOP, K., & KERDPRASOP, N. An empirical study of distance metrics for k-nearest neighbor algorithm. International Conference on Industrial Application Engineering, pp. 280–285, 2015.

DING, L., YU, F., PENG, S., & XU, C. A classification algorithm for network traffic based on improved support vector machine. Journal of Computers, 8, 2013

EDLA, D., MANGALOREKAR, K., DHAVALIKAR, G., & DODIA, S. 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, 2018.

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

FARHADY, H., LEE, H., & NAKAO, A. Software-defined networking: A survey. Computer Networks, 81, pp. 79–95, 2015.

FERREIRA, F. R. O uso de rede neural artificial mlp na predição de estruturas secundárias de proteínas. Dissertação de mestrado, Universidade Estadual Paulista, Instituto de Biociências, Letras e Ciências Exatas, São José do Rio Preto, 2004.

FIRMINO, M. Testes de hipóteses: uma abordagem não paramétrica. Dissertação de mestrado, Universidade de Lisboa, Faculdade de Ciências, Departamento de Estatística e Investigação Operacional, 2015.

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

GARCÍA-PEDRAJAS, N., ROMERO DEL CASTILLO, J., & CERRUELA-GARCÍA, G. A proposal for local k values for k -nearest neighbor rule. IEEE Transactions on Neural Networks and Learning Systems, 28(2), pp. 470–475, 2017.

JEDARI, E., WU, Z., RASHIDZADEH, R., & SAIF, M. Wi-fi based indoor location positioning employing random forest classifier. International Conference on Indoor Positioning and Indoor Navigation (IPIN), pp. 1–5, 2015.

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

MAIA, N. A. Engenharia de Tráfego em Domínio MPLs utilizando Técnicas de Inteligência Computacional. Tese de doutorado, Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brasil, 2006.

PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., MICHEL, V., THIRION, B., GRISEL, O., . . . DUCHESNAY, E. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, pp. 2825–283, 2011.

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

SINGH, A., HALGAMUGE, M., & LAKSHMIGANTHAN, R. 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), 2017.

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

VAPNIK, Vladimir Naumovich et al. Statistical learning theory. 1998.

WU, Z., XU, Q., LI, J., FU, C., XUAN, Q., & XIANG, Y. Passive indoor localization based on csi and naive bayes classification. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 48, pp. 1566–1577, 2018.