10.18605/2175-7275/cereus.v13n3p14-30

ARTIGO ORIGINAL

<<Recebido em: 19/04/2021 Aceito em: 16/06/2021. >>

REVISTA CEREUS ISSN: 2175-7275

Classificação de tráfego entrante em uma topologia SDN

Classification of incoming traffic in an SDN topology

Victor de Freitas Arruda[1], Nilton Alves Maia[2], Maurílio José Inácio[3]

RESUMO

As redes de computadores se transformaram em uma ferramenta vital para o transporte de informações. O uso de Redes Definidas por Software pode viabilizar o desenvolvimento de técnicas para melhorar o desempenho das redes com relação à segurança, qualidade de serviço e engenharia de tráfego. A implementação dessas técnicas pode ser facilitada com a classificação do tráfego entrante na rede. Este trabalho propôs 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.

Palavras-chave: Machine Learning, Classificação de Tráfego, Redes Definidas por Software

ABSTRACT

Computer networks have become a vital tool for transporting information. The use of Software Defined Networks can enable the development of techniques to improve network performance with respect to security, quality of service and traffic engineering. The implementation of these techniques can be facilitated by classifying incoming traffic on the network. This work proposed a comparative study of machine learning algorithms for the classification of incoming traffic in an SDN topology. The classifiers' performance was assessed using the metrics accuracy, precision, recall and f1-score, in addition to the training and validation times of the models. The Random Forest algorithm was considered the most efficient in the considered traffic classification scenario. He achieved values similar to the best results regarding the metrics accuracy, precision, recall and f1-score, but obtained lower values in the training and validation times.

Key-Words: Machine Learning, TrafficClassification, Software Defined Networks

[1] Mestrando do Programa de Pós-Graduação em Modelagem Computacional e Sistemas (PPGMCS), UNIMONTES.

E-mail:

victorfarruda@outlook.com

[2] Doutor em Engenharia Elétrica, UFMG, Professor do PPGMCS.

[3] Doutor em Engenharia Elétrica, UFMG, Professor do PPGMCS.

I. 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 et al., 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 et al., 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 et al., 2016).

As Redes SDN são um novo paradigma em telecomunicações e redes de computadores. As redes SDN 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 o 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 (comutadores compatíveis com SDN) e o controlador, ou seja, o controlador usa a interface *southbound* dos dispositivos comutadores habilitados para SDN para conectar-se ao plano de dados. Já o plano de controle atua como “cérebro” da rede. O plano de controle 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, 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 (KIM E FEAMSTER, 2013; FARHADY, LEE, E 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 serviço (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.

Para BAKKER et al.(2019), 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. O objetivo da classificação de tráfego é melhorar o gerenciamento de recursos de rede, segurança 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 QoS(FAN E LIU, 2017).A classificação de tráfego, pode ser efetuada utilizando números de porta,*payload*e técnicas de*machinelearning*.

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 E LIU, 2017; AMARAL et al., 2016).

A classificação baseada em inspeção de *payload*, também conhecida por*DeepPacketInspection*(DPI), é atualmente uma das técnicas mais utilizadas(AMARAL et al., 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, às 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 DPI resulta em altos custos computacionais e requer manutenção manual de assinaturas (FAN E LIU, 2017).

Os métodos de classificação baseados em *machinelearning* podem superar algumas das limitações de abordagens baseadas em porta e *payload*. Mais especificamente, as técnicas de *machinelearning* 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 E 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 et al., 2016). Os classificadores utilizados neste trabalho utilizam o aprendizado supervisionado. Mais especificamente, são utilizados modelos derede neural artificial (RNA) do tipo MLP(*MultilayerPerceptron)* utilizando o ADAM e o LBFGS(*Limited-MemoryBroyden-*

Fletcher-Goldfarb-Shanno), oSVM*(Support Vector Machine), NaiveBayes, KNN(K-NearestNeighbor) e o Random Forest*.

As RNAsdo tipo MLP fazemcom 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 na rede neural, pois é comprovado que sem estes é impossível a resolução de problemas linearmente não separáveis (FERREIRA, 2004). Nesse tipo de rede, cada uma das camadas tem sua função específica. Assim, cada neurônio calcula uma soma ponderada das entradas e passa essa soma na forma de uma função não-linear limitada. Em nível de mesoestrutura, tem-se duas ou mais camadas com conexão *feedforward*(VIEIRA E BAUCHSPIESS, 1999). Pode-se afirmar ainda que com a adição de uma ou mais camadas intermediárias(ocultas), o poder computacional de processamento não-linear e de armazenamento da rede é aumentado. Em uma camada oculta grande o suficiente, é possível fazer a representação de qualquer função contínua das entradas. As saídas dos neurônios de cada camada intermediária são utilizadas como entrada para a camada posterior.

As SVMs foram desenvolvidas por Vapnik (1995) a partir dos estudos de VAPNIK E CHERVONENKIS (1971); e BOSER et al.(1992). A classificação através da SVM 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 (HAYKIN, 2001).

O *NaiveBayes* é um algoritmo de *machinelearning*, onde o classificador aprende por meio de um algoritmo de classificação de documentos (BUŽIĆ E DOBŠA, 2018). O classificador *NaiveBayes* se baseia em duas suposições básicas: 1) as características são independentes umas das outras e 2) cada característica tem a mesma proeminência(WU, et al., 2018). O classificador *NaiveBayes* 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 et al., 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 (BUŽIĆ E DOBŠA, 2018).

O *K-NearestNeighbor*(KNN) é um algoritmo que utiliza agrupamento para efetuar a classificação dos dados, dessa forma ele pode utilizar aprendizado supervisionado ou não supervisionado para efetuar a classificação. Neste 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 et al., 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, CASTILLO, E 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, E 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 et al., 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, E 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 et al., 2015).

Este trabalho propõe a implementação de um sistema de classificação do tráfego de aplicações entrantes em uma topologia SDN. As aplicações são Voip, Telnet, Quake3, DNS, CSi, CSa, Vídeo e Web. Para executar a classificação foram comparados os modelos de RNA do tipoMLP(utilizando o ADAM e o LBFGS), oSVM, o KNN, o *NaiveBayes* e o *Random Forest*.

As próximas seções deste artigo são organizadas da seguinte maneira: na seção 2 é apresentada a metodologia. Na seção 3 são apresentados os resultados obtidos com os

experimentos. Na seção 4 são discutidos os resultados obtidos, e em seguida, são apresentadas as conclusões sobre o trabalho.

II. METODOLOGIA

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

Figura 1. Metodologia utilizada no trabalho

A simulação da topologia SDN foi realizada com a utilização do software Mininet(MININET TEAM, 2020) executado na plataforma SND Hub(SDN HUB, 2020). O código para implementação das topologias SDN no Mininet foi escrito em linguagem *Python*. O software D-ITG (*Distribuited Internet TrafficGenerator*) (BOTTA, DAINOTTI&PESCAPÈ, 2012) foi utilizado para gerar o tráfego das aplicações a partir dos hosts. Para a captura do tráfego foi utilizado o tcpdump. Após a captura do tráfego, o tcpdump gera um arquivo no formato pcap, este arquivo é transformado em csv pelo tshark. O tshark é uma implementação do Wireshark em modo texto, muito similar ao tcpdump, porém permite a conversão do arquivo pcap para csv(utilizado neste trabalho).

As etapas utilizadas para a execução da medição do tráfego são descritas na Figura 2. O primeiro passo é inicialização do controlador *floodlight*. Em seguida a topologia SDN é iniciada no software Mininet. Além disso, a aplicação *tcpdump* começa a capturar o tráfego que trafega pelas interfaces da topologia SDN. Logo em seguida, oD-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 programa de medição (promed) processa o arquivo csv e gera a medição de tráfego.

Na topologia de rede SDN considerada 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 à qualquer máquina que esteja rodando o módulo servidor. Quake3 é um jogo

eletrônico de tiro em primeira pessoa. 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.

Figura 2. Etapas para execução da medição de tráfego

No promed foi utilizada a biblioteca pandas (THE PANDAS DEVELOPMENT TEAM, 2020) e o *spark*(COMMITTERS, 2021) 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. Ao final do processamento foi gerado um novo arquivo e esse arquivo de medição é utilizado como entrada nos 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 m*achinelearning* 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 classes de medição de tráfego identificadas. Os algoritmos das RNAs do tipo MLP (ADAM e LBFGS), SVM, KNN, *NaiveBayes* e *Random Forest* 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 configuradas 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 BasisFunction*).

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 *NaiveBayes,* 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.

III. RESULTADOS

Nesta seção são apresentados os resultados da avaliação dos algoritmos selecionados para classificação do trafego entrante na topologia SDN. Foi utilizada uma topologia SDN adaptada a partir do trabalho de MAIA (2006). Os algoritmos foram avaliados considerando-se a Acurácia, Precisão, Revocação, F1-Score, tempo de treinamento e tempo de validação do modelo. Cada um dos algoritmos de classificação foi executado 100(cem) vezes.

A seleção dos melhores parâmetros para os algoritmos selecionados no cenário 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.

A topologia SDN utilizada é apresentada na Figura 3. 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.

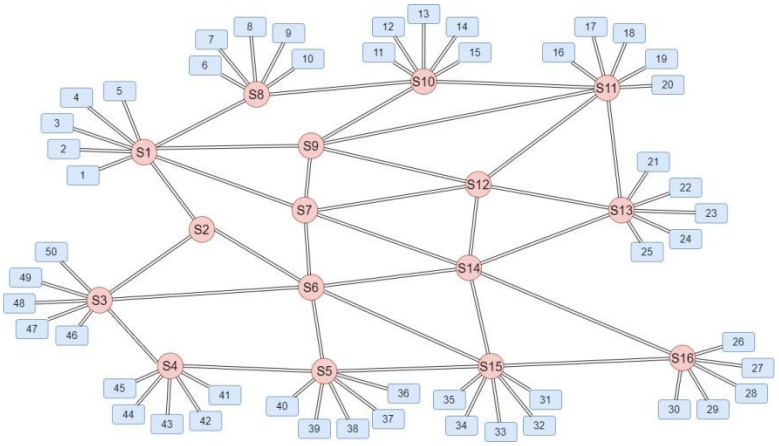

Figura 3. Topologia SDN (Adaptado de MAIA, 2006)

Na Figura 4, observa-se o gráfico de caixa da acurácia dos algoritmos selecionados na topologia SDN utilizada. Pode-se verificar que o algoritmo SVM obteve os melhores resultados com média de 91,30%, seguido do *Random Forest* com média de 91,26%, a rede MLP utilizando o algoritmo LBFGScom média de 91,07%, a rede MLP utilizando o algoritmo ADAM com média de 90,99%, KNN com média de 89,51% e *NaiveBayes* com média de 74,49%. Além disso, nota-se que, com exceção do *NaiveBayes*, todos os algoritmos tiveram um baixo desvio padrão.

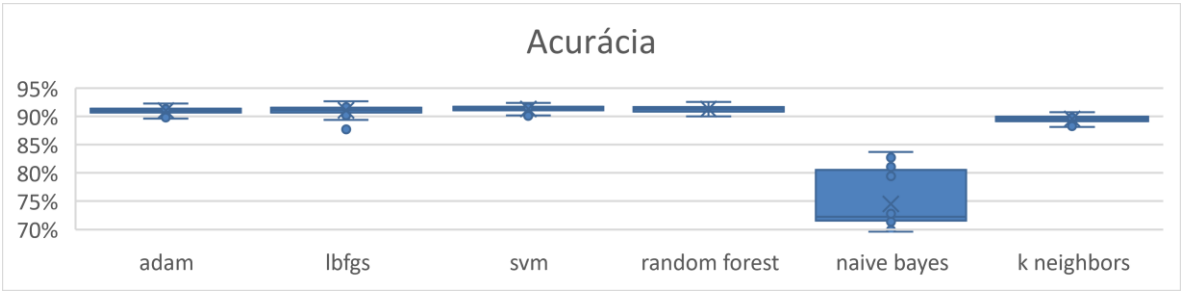

Figura 4. Acurácia dos algoritmos na topologia SDN utilizada

Na Figura 5, observa-se o gráfico de caixa de precisão dos algoritmos selecionados na topologiaSDN utilizada. Pode-severificar que a rede MLP utilizando o algoritmo LBFGS obteve melhores resultados com média de 92,17%, seguido do algoritmo SVM com média

de 91,98%, a rede MLP utilizando o algoritmo ADAM com média de 91,77%, *Random Forest* com média de 91,35%, KNN com média de 89,59% e *NaiveBayes* com média de 68,19%. Além disso, nota-se que, com exceção do *NaiveBayes*, todos os algoritmos tiveram um baixo desvio padrão.

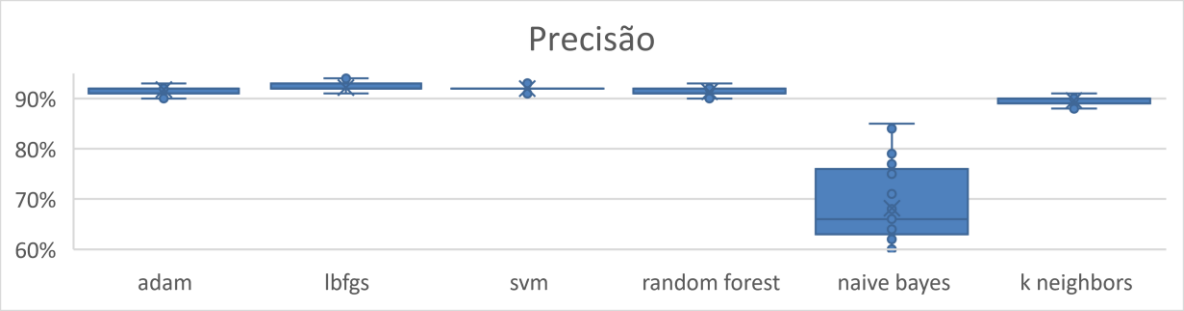

Figura 5. Precisão dos algoritmos na topologia SDN utilizada

Na Figura 6, observa-se o gráfico de caixa da revocação dos algoritmos selecionados na topologia SDN utilizada. Pode-se verificar que o algoritmo SVM obteve melhores resultados com média de 91,29%, seguido do *Random Forest* com média de 91,25%, a rede MLP utilizando o algoritmo LBFGScom média de 91,09%, a rede MLP utilizando o algoritmo ADAM com média de 90,98%, KNN com média de 89,51% e *NaiveBayes* com média de 74,51%. Além disso, nota-se que, com exceção do *NaiveBayes*, todos os algoritmos tiveram um baixo desvio padrão.

Figura 6. Revocação dos algoritmos na topologia SDN utilizada

Na Figura 7, observa-se o gráfico de caixa da f1-score dos algoritmos selecionados na topologia SDN utilizada. Pode-se verificar que o algoritmo SVM obteve os melhores resultados com média de 91,36%, seguido do *Random Forest* com média de 91,25%, a rede MLP utilizando o algoritmo LBFGScom média de 91,28%, a rede MLP utilizando o algoritmo ADAM com média de 91,15%, KNN com média de 89,51% e *NaiveBayes* com média de 68,98%. Além disso, nota-se que, com exceção do *NaiveBayes*, todos os algoritmos tiveram um baixo desvio padrão.

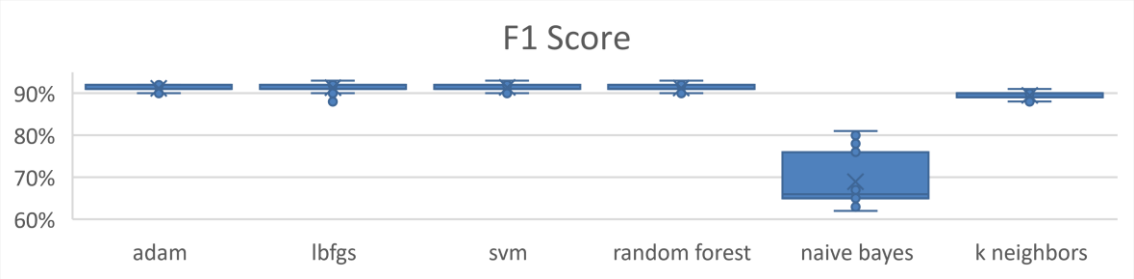

Figura 7.F1-Score dos algoritmos na topologia SDN utilizada

Na Figura 8, observa-se o gráfico de caixa do tempo de treinamento dos algoritmos selecionados na topologia SDN utilizada. Pode-se verificar que o algoritmo KNN obteve melhores resultados com média de 0,0071 segundos,seguido do *NaiveBayes* com média de 0,0068 segundos, *Random Forest*com média de 0,0510 segundos, SVM com média de2,3559 segundos, a rede MLP utilizando o algoritmo LBFGScom média de 8,6103 segundos e a rede MLP utilizando o algoritmo ADAMcom média de 20,2700 segundos. Além disso, nota-se que todos os algoritmos obtiveram um baixo desvio padrão.

Na Figura 9, observa-se o gráfico de caixa do tempo de validação dos algoritmos selecionados na topologia SDN utilizada. Pode-se verificar que o algoritmo *NaiveBayes*obteve melhores resultados com média de 0,0010 segundos,seguido do *Random Forest*com média de 0,0025 segundos, a rede MLP utilizando o algoritmo LBFGScom média de 0,0035 segundos, KNN com média de 0,0088 segundos, a rede MLP utilizando o algoritmo ADAMcom média de 0,0088 segundos, SVM com média de 0,0364 segundos. Além disso, nota-se que todos os algoritmos obtiveram um baixo desvio padrão.

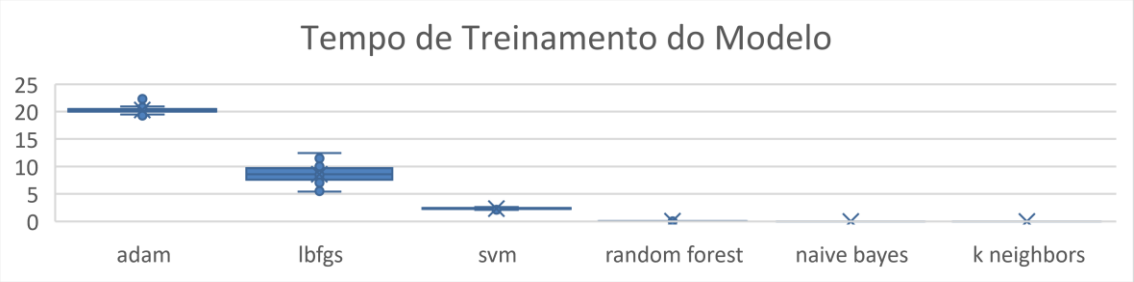

Figura 8. Tempo de treinamento dos algoritmos na topologia SDN utilizada

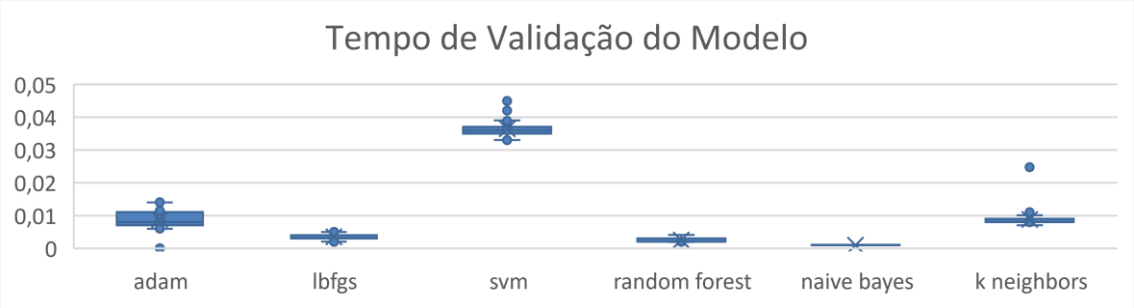

Figura 9. Tempo de validação dos algoritmos na topologia SDN utilizada

IV. DISCUSSÃO

Com o objetivo de comparar o desempenho dos classificadores de tráfego utilizados neste trabalho, apresenta-se na Tabela 1, um resumo dos resultados obtidos pelos algoritmos *Random Forest*, *NaiveBayes*, KNN, SVM e MLP com ADAM e LBFGS durante as 100(cem) execuções realizadas.

Tabela 1 - Valores médios da Acurácia, Precisão, Revocação, F1-Score, Tempo de treinamento e Tempo de validação dos algoritmos *Random Forest*, *NaiveBayes*, KNN, SVM e MLP com ADAM e L-BFGS.

	Random Forest	*NaiveBayes*	KNN	SVM	MLP com ADAM	MLP com LBFGS
Acurácia	91,26%	74,49%	89,51%	91,30%	90,99%	91,07%
Precisão	91,35%	68,19%	89,59%	91,98%	91,77%	92,17%
Revocação	91,25%	74,51%	89,51%	91,29%	90,98%	91,09%
F1-Score	91,25%	68,98%	89,51%	91,36%	91,15%	91,28%
Tempo de treinamento	51,0ms	6,8ms	7,1ms	2355,9ms	20270ms	8610,3ms
Tempo de validação	2,5ms	1,0ms	8,8ms	36,4ms	8,8ms	3,5ms

Observando-se os resultados apresentados na Tabela 1, pode-se afirmar que o SVM obteve os melhores resultados com relação a Acurácia, Revocação e F1-Score, enquanto que a RNA do tipo MLP utilizando LBFGS obteve melhor resultado na Precisão. Por outro lado, o *NaiveBayes*obteve o melhor resultado com relação ao Tempo de treinamento e Tempo de validação e os piores resultados com relação a Acurácia, Precisão, Revocação e F1-Score.

Analisando-se novamente os resultados da Tabela 1, pode-se observar que os resultados do *Random Forest*com relação a Acurácia, Revocação e F1-Score são bastante próximos ao SVM, apesar de inferiores. No caso da Precisão, apesar de ser também

inferior, o *Random Forest* apresenta um resultado próximo ao da RNA do tipo MLP utilizando LBFGS. Por outro lado, considerando os Tempos de treinamento e Validação pode-se notar que o algoritmo *Random Forest* apresenta melhores resultados que o SVM e RNA do tipo MLP utilizando LBFGS. Comparando-se com o SVM, pode-se observar que o Tempo de treinamento do *Random Forest* é aproximadamente quarenta e seis (46) vezes menor, enquanto o Tempo de Validação é quatorze (14) vezes menor. No caso da comparação com a RNA do tipo MLP utilizando LBFGS, pode-se observar que o Tempo de treinamento do *Random Forest* é aproximadamente cento e sessenta e oito (168) vezes menor, enquanto o Tempo de Validação é 1,4 vezes menor. Poder-se-ia argumentar que os tempos de treinamento e Validação do*Random Forest* são piores que o *NaiveBayes*. Ocorre que o *NaiveBayes* apresenta resultados de classificação muito inferiores com relação a Acurácia, Precisão,Revocação e F1-Score.

Portanto, pode-se afirmar que o algoritmo*Random Forest* é o mais eficiente no cenário de classificação de tráfego considerado. Entretanto, para uma melhor avaliação dos resultados seria necessário aplicar algum método de avaliação estatística e comparar com outra(s) topologia(s) de rede.

V. CONCLUSÕES

Este trabalho teve como objetivo realizar o estudo comparativo de algoritmos de *machinelearning* aplicados a classificação de tráfego em uma topologia de rede SDN. Utilizou-se a plataforma Mininet e a linguagem Python para construção da topologia 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 e processados pelo programa de medição foi construído um arquivo com os padrões de treinamento e validação que foram utilizados pelos algoritmos de *machinelearning*. 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.

Como trabalhos futuros pretende-se adicionar mais formas de classificação (classificadores do tipo ensemble), visando fazer a comparação com os algoritmos utilizados neste trabalho e avaliar os resultados utilizando algum método de avaliação

estatística para assim justificar melhor os resultados obtidos. Pretende-se também fazer um estudo de caso para verificar a eficácia do melhor balanceamento de carga com base nos dados gerados pelos algoritmos de *machinelearning*.

REFERÊNCIAS

AMARAL, P.; DINIS, J.;PINTO, P.; BERNARDO, L.; TAVARES, J.; MAMED, H. S. Machine Learning in Software Defined Networks: Data Collection and Traffic Classification. **IEEE 24th International Conference On Network Protocols(ICNP***)*, 2016. Disponível em: https://ieeexplore.ieee.org/document/7785327. Acesso em: 02 mar. 2021.

BAKKER, J.; NG, B.; SEAH, W. K.; PEKAR, A. Traffic Classification with Machine Learning in a Live Network. **FIP/IEEE International Symposium on Integrated Network Management (IM2019)**, p.488-493, 2019. Disponível em: https://ieeexplore.ieee.org/abstract/document/8717890. Acesso em: 02 mar. 2021.

BISOL, R. V.;SILVA, A. S.;MACHADO, C. C.;GRANVILLE, L. Z.; SCHAEFFER-FILHO, A. E. 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. Disponível em: http://www.inf.ufrgs.br/~granville/wp-content/papercite-data/pdf/152271.pdf. Acesso em: 03 mai. 2021.

BOSER, B. E.;GUYON, I. M.;VAPNIK, V. N. A training algorithm for optimal margin classifiers. **Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory**, p. 144-152, 1992.

BOTTA, A.; DAINOTTI, A.; PESCAPÈ, A. A tool for the generation of realistic network workload for emerging networking scenarios. ***Computer Networks (Elsevier),*** *p.* 3531-3547, 2012.

BUŽIĆ, D.; DOBŠA, J. Lyrics Classification using Naive Bayes. ***International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO),*** p. 21-25,2018.

CARDOSO, W. S.; SILVA, F. J.; ROCHA, U. V.; SOUSA, M. P. Implantação de um Patch Panel Virtual Utilizando Redes Definidas por Software. ***Anais do XXIII Simpósio Brasileiro de Sistemas Multimídia e Web: Workshops e Pôsteres,*** p. 95-98, 2017.

CHOMBOON, K.; CHUJAI, P.; TEERARASSAMEE, P.; KERDPRASOP, K.; KERDPRASOP, N. An Empirical Study of Distance Metrics for k-Nearest Neighbor Algorithm. ***International Conference on Industrial Application Engineering*****,** p. 280-285, 2015.

COMMITTERS, A.*Apache Spark: motor de análise unificado ultrarrápido. Motor de análise unificado ultrarrápido*. **Página Web.** Disponível em: https://spark.apache.org. 2021. Acesso em: 02 mar. 2021.

DING, L.; YU, F.; PENG, S.; XU, C. A Classification Algorithm for Network Traffic based on Improved Support Vector Machine. ***Journal of Computers,*** p. 1090-1096, 2013.

EDLA, D. R.; MANGALOREKAR, K.; DHAVALIKAR, G.; DODIA, S. Classification of EEG data for human mental state analysis using Random Forest Classifier. ***Procedia Computer Science,*** p. 1523-1532, 2018.

FAN, Z.; LIU, R. Investigation of machine learning based network traffic classification. ***International Symposium On Wireless Communication Systems(ISWCS)*****,** 2017.

FARHADY, H.; LEE, H.; NAKAO, A. Software-Defined Networking: A survey. ***Computer Networks,*** p. 79–95, 2015.

FERREIRA, F. R. *O uso de rede neural artificial MLP na predição de estruturas secundárias de proteinas.* 2004. 76p. Dissertação de Mestrado. Universidade Estadual Paulista, Instituto de Biociências, Letras e Ciências Exatas, São José do Rio Preto.

GARCÍA-PEDRAJAS, N.; CASTILLO, J. A.; CERRUELA-GARCÍA, G. A Proposal for Local k Values for k-Nearest Neighbor Rule. ***IEEE Transactions on Neural Networks and Learning Systems 1,*** p. 470-475, 2017.

HAYKIN, S. *Redes Neurais: princípios e práticas.* 2 ed. Porto Alegre: Bookman, 2001. 900p.

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)*****,** p.1-5, 2015.

KIM, H.; FEAMSTER, N.Improving network management with software defined. ***IEEE Communications Magazine*****,** p. 114–119, 2013.

KOKILA, R.; SOMASUNDARAM, T. S.; KANNAN, G. DDoS detection and analysis in SDN-based environment using support vector machine classifier. **2014 Sixth International Conference on Advanced Computing (ICoAC)**, **p. 205-210, 2014.** Disponível em: https://ieeexplore.ieee.org/abstract/document/7229711. Acesso em: 02 mar. 2021.

MAIA, N. A.*Engenharia de Tráfego em Domínio MPLS utilizando Técnicas de Inteligência Computacional.* 2006. 192p. *Tese de Doutorado. Programa de Pós-Graduação em Engenharia Elétrica da Universidades Federal de Minas Gerais (PPGEE). Disponível em: https://repositorio.ufmg.br/handle/1843/BUOS-8CLE75.* Acesso em: 02 mar. 2021.

MININET TEAM. *Mininet: An Instant Virtual Network on your Laptop (or other PC) - Mininet.* ***Documentação*****.** Disponível em: http://mininet.org. Acesso em: 02 mar. 2020.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.;BLONDEL, M.; PRETTENHOFER P.; WEISS, R.; DUBOURG,V.; VANDERPLAS,J.; PASSOS,A.; COURNAPEAU,D.; BRUCHER,M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine Learning in Python. ***Journal of Machine Learning Research 12*****,** p. 2825-2830, 2011.

SDN HUB. All-in-one SDN App Development Starter VM | SDN Hub. Disponível em: http://sdnhub.org/tutorials/sdn-tutorial-vm. Acesso em: 02 mar. 2020.

SINGH, A.; HALGAMUGE, M. N.; 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,*** 2017.

TEAM, THE PANDAS DEVELOPMENT.*Pandas - Python Data Analysis Library*. Disponível em: https://pandas.pydata.org/. Acesso em: 02 mar. 2021.

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)*****,** p. 571-576, 2013.

VAPNIK, V. N. *The Nature of Statistical Learning Theory.* New York, 1995. 314p.

VAPNIK, V. N.*Statistical Learning Theory,* 1998. 768p.

VAPNIK, V. N.; CHERVONENKIS, A. Y. On the uniform convergence of relative frequencies of events to their probabilities. ***Theory of Probability and its Applications,*** p. 264-280, 1971.

VIEIRA, Z. P.; BAUCHSPIESS, A. Implementação do Servocontrole Auto-Sintonizado em Tempo-Real Utilizando Rede Perceptron Multicamadas. ***IV Congresso Brasileiro de Redes Neurais*****,** p. 308-313, 1999.

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*****,** p. 1566-1577, 2018.