UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e ingeniería

Machine Learning Based Traffic Steering for Software Defined Networks

Marlon Sebastián Rodríguez Flor

Ingeniería en Ciencias de la Computación

Trabajo de fin de carrera presentado como requisito
para la obtención del título de
Ingeniero en Ciencias de la computación

Quito, 6 de mayo de 2021

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e ingeniería

HOJA DE CALIFICACIÓN
DE TRABAJO DE FIN DE CARRERA

Machine Learning Based Traffic Steering for Software Defined Networks

Marlon Sebastián Rodríguez Flor

Nombre del profesor, Título académico **Noel Pérez, Ph.D.**

Quito, 6 de mayo de 2021

© DERECHOS DE AUTOR

Nombres y apellidos: Marlon Sebastián Rodríguez Flor

Código: 00140141

Cédula de identidad: 1721099370

Lugar y fecha: Quito, 6 de mayo de 2021

ACLARACIÓN PARA PUBLICACIÓN

Nota: El presente trabajo, en su totalidad o cualquiera de sus partes, no debe ser considerado como una publicación, incluso a pesar de estar disponible sin restricciones a través de un repositorio institucional. Esta declaración se alinea con las prácticas y recomendaciones presentadas por el Committee on Publication Ethics COPE descritas por Barbour et al. (2017) Discussion document on best practice for issues around theses publishing, disponible en http://bit.ly/COPETheses.

UNPUBLISHED DOCUMENT

Note: The following capstone project is available through Universidad San Francisco de Quito USFQ institutional repository. Nonetheless, this project – in whole or in part – should not be considered a publication. This statement follows the recommendations presented by the Committee on Publication Ethics COPE described by Barbour et al. (2017) Discussion document on best practice for issues around theses publishing available on http://bit.ly/COPETheses.

RESUMEN

En este trabajo se propone utilizar modelos de aprendizaje automático supervisado para dirigir el tráfico en una red definida por software. El objetivo es balancear el tráfico considerando el ancho de banda de los enlaces existentes en la red. En este trabajo se creó una red virtual, para la cual se desarrolló una aplicación de control para direccionar el tráfico por diferentes enlaces. Considerando el funcionamiento convencional, se generó tráfico en la red para construir datasets y alimentar los modelos de aprendizaje automático. Específicamente, se crean 2 datasets y los modelos utilizados son: redes neuronales artificiales, Support Vector Machines, k-nearest neighbors, para los cuales se provee una configuración empírica. Todos los modelos usados en ambos datasets obtienen excelentes resultados, sin embargo, debido a que no se obtuvo una diferencia significativa en las métricas de los modelos, no se pudo determinar a un modelo como ganador. Adicionalmente, solo para el segundo dataset los modelos fueron capaces de generar alternabilidad en los paths de la red SDN, alcanzando el comportamiento deseado.

Palabras clave: SDN, Machine Learning, Artificial Neural Networks, Support Vector Machine, k-Nearest Neighbors.

ABSTRACT

In this work we propose to use supervised machine learning models to direct traffic in a software-defined network. The objective is to balance the traffic considering the bandwidth of the existing links in the network. In this work, a virtual network was created, for which a control application was developed to direct traffic through different links. Considering the conventional operation, traffic was generated on the network to build datasets and feed machine learning models. Specifically, two datasets were created, and the models used are: artificial neural networks, support vector machines, and k-nearest neighbors, for which empirical configurations are provided. All the models used in both datasets obtained excellent results, however, since a significant difference was not obtained in the metrics of the models, a winning model could not be determined. Additionally, only for the second dataset the models were able to generate alternation in the paths of the SDN network, reaching the desired behavior.

Key words: SDN, Machine Learning, Artificial Neural Networks, Support Vector Machine, k-Nearest Neighbors.

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCTION

The constant and rapid growth of the Internet in recent years has contributed to the complexity of communication networks, making them more diverse and extensive, due to the arrival of new infrastructure such as 5G networks or the internet of things (IoT) (Cai et al., 2018; Fadlullah et al., 2017; Gubbi et al., 2013). Because of the large traffic that flows through the network and its complexity, it is difficult to monitor and maintain it. According to Cisco, for 2021 a global IP traffic of 3.3 Zettabytes was expected, but, considering the COVID-19 pandemic, there has been an increase in packet loss, congestion, and internet latency variability (Candela et al., 2020; Forecast & G. M. D. T, 2019). For this reason, the traffic could be even higher than expected by Cisco for this year.

Some approaches have been proposed to improve the quality of service (QoS) in communication networks. The current proposals are focused on machine learning classifer (MLC) based techniques, which have been used to classify and predict network traffic. According to the state of the art Decision Trees (DT), support vector machines (SVM), Naive Bayes (NB), and artificial neural networks (ANN) are frequently used techniques. For example, Shafiq et al. (2016) use the following models C4.5 DT, SVM, BayesNet, and NB classifiers, reaching the C4.5 DT the higher classification accuracy (ACC) score of 78.91%. Michael, A, Valla, Neggatu, and Moore (2017) use a multilayer perceptron ANN classifier with 248 and 12 neurons for the input and output layers, respectively. In addition, the proposal includes one hidden layer with 12 neurons. This model obtained a classification ACC score of 97%. Moore and Zuev (2005) employed two configurations of the NB classifier with and without kernel density estimation. Additionally, a fast correlation-based filter was employed together with both NB classifiers, obtaining ACC scores of 93.38% and 93.73%, respectively. Recently, a deep neural network with two hidden layers with 500 and 100 neurons achieved a classification ACC score of 98.5% (Shu, Jiang, & Sun, 2018).

Although the use of MLCs has been proven to be successful in the classification and prediction of network traffic on traditional networks, there is no evidence reported in the literature of their use on modern software-defined networks (SDN). This type of network is characterized by having separate control and data plane, making them highly scalable, dynamic, and programmable (Azodolmolky, 2013). Therefore, in this work, we propose using three supervised learning MLCs to address the problem of classification and prediction of traffic in modern SDN.

This paper is organized as follows: the materials and methods section present the definition of the SDN, the datasets creation, and a brief description of selected MLCs; The results and discussion section argue the obtained performance of employed MLCs. Finally, the conclusions and future work are reported.

MATERIALS AND METHODS

Software defined networks (SDN)

A SDN is a new paradigm in networks, in which the control and data plane are decoupled, which offers multiple benefits such as: programmable infrastructure, centralized administration, dynamic optimization and automation (Azodolmolky, 2013). The control plane refers to a software which is responsible for having data forwarding policies such as routing tables, topology tables or Address Resolution Protocol (ARP) tables, among others. While the data plane is the network device that is responsible for forwarding the data.

The SDN controller oversees managing the network infrastructure, through the communication with a southbound protocol such as OpenFlow, allowing a centralized administration as shown in Figure *1*. Regarding, the SDN controller, it is worth mentioning the wide array of commercial and open-source solutions that exist. In this context, considering that machine models are mainly implemented with Python's libraries throughout this work, the Ryu SDN controller is selected. Ryu controller is impolemented in Python and provides a rich northbound API that simplified the development of network applications.

Network topology

We considered the creation of a virtualized network using the virtual networks over Linux (VNX) virtualization tool, which is designed to build virtual networks for testbeds (Fernández et al., 2012). Additionally, the network follows the SDN paradigm, so Open vSwitch devices (OVS) are used as the data plane (Pfaff et al., 2015). Later, these devices will be controlled by a network application running on Ryu.

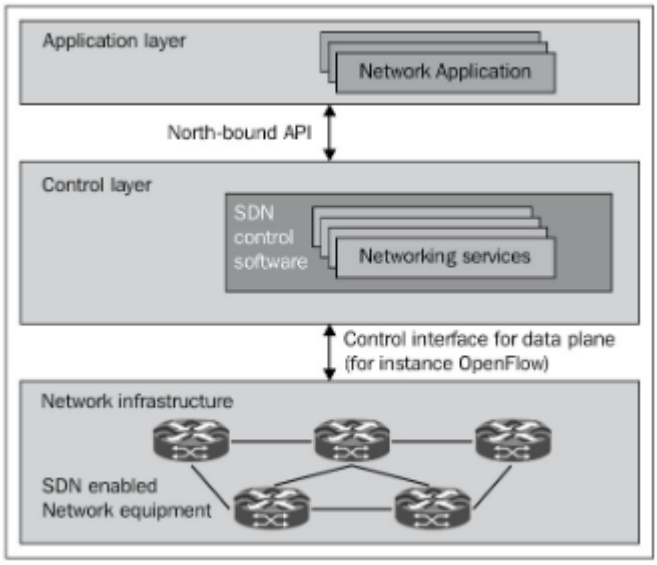

Figure 1: The SDN architecture (figure taken from (Azodolmolky, 2013))

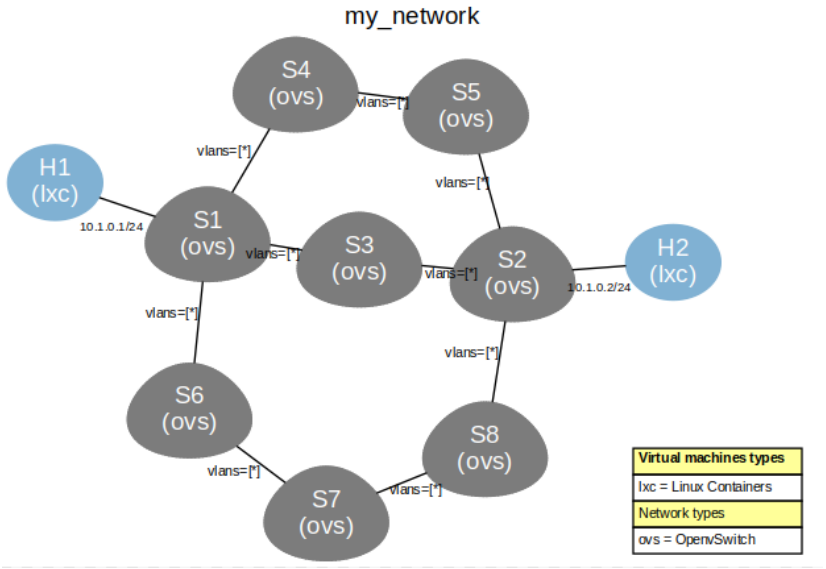

Figure 2: Implemented network topology.

The implemented network topology consists of 8 OVS nodes with limited 10 Mbps bandwidth per link. It can be distinguished as a partial mesh topology, i.e., an OVS node is connected to all the remaining devices or some of them that are linked to the others (Davies, 2019), as shown in Figure *2*. Besides, two machines H1 and H2, were added upon S1 and S2 networks to guarantee three communication paths between both machines. This structure allows simulating a real production network.

Experimental datasets

We used the aformentioned described network topology (see Figure *2*) to create two different experimental datasets named **Dataset1** and **Dataset2**.

The **Dataset1** considered the following parameters: datapath id (OVS), source, input port, bandwidth, packets sent and received, bytes sent and received, and path. Since the topology in Figure *2* has several paths, the Spanning Tree Protocol (STP) was used in order to obtain data on all paths. STP operates on layer 2 of the OSI model, its function is to find loops in network topologies and enable or disable links to eliminate these loops. In our case, the STP is used in a different way, ports (links) were closed intentionally to force the STP to find new paths in the topologies, obtaining data from the network traffic in all paths. Moreover, to simulate the traffic through the network, the Iperf tool was used mainly, which allows generating measurements of the maximum achievable bandwidth (Silva & Alves, 2014). Specifically, the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) protocols were used, which operate in the transport layer of the OSI model. Additionally, the Netem tool was used to generate errors in the sent packages.

On the other hand, the **Dataset2** employed the datapath id (OVS), source, input port, chosen path, path-1, path-2, and path-3 bandwidths parameters. Specifically, this dataset differs from the **Dataset1** in that for each observation the bandwidth on each path is taken into account and the path chosen corresponds to the path with the less congestion. Similarly, the Iperf tool was used to generate traffic for the network. When the last bandwidth of the chosen path is greater than 5 Mbps, the path with the lowest bandwidth is searched and it is chosen, in order to alleviate the traffic on the network.

Machine learning classifiers

Artificial neural networks.

Artificial Neural Networks (ANN) are supervised learning models inspired by brain neural networks. An ANN is made up of layers of neurons, at least one input and output layer, the neurons of one layer connect to the neurons of the layer that precedes it, forming connections (Lantz, 2015). Each of the neurons has a weight, which is adjusted as the algorithm progresses, through an algorithm called Backpropagation. This method seeks to minimize a loss function, for this the algorithm is repeated several times minimizing the error and adjusting the weights.

Support vector machine.

Support vector machines (SVM) are very powerful supervised learning classifiers that separates similar classes with the use of hyperplanes, obtaining a maximum margin between these classes (Lantz, 2015). A SVM model is capable of separating classes that are not linearly separable with the use of kernerls. A Kernel is a function that maps data in a higher dimensional space, obtaining more characteristics on the data, where the algorithm is able to separate two classes with a hyperplane. This is possible with the use of a process known as the kernel trick, which allows operating in the upper dimensional space without calculating the coordinates of the data in this space, making it a much less expensive computational process.

K-nearest neighbors.

The k-nearest neighbors (kNN) algorithm is one of the simplest supervised learning classifiers. Its name is due to the fact that it uses one or more k-nearest instances to classify unlabeled examples (Lantz, 2015). This method is based on the similarity, which is determined by the Euclidean distance, a number k of most similar neighbors is taken into account to label an unlabeled observation.

Experimental setup

Data processing.

For *Dataset1* and *Dataset2* the same data treatment was used. The categorical variables were encoded with the OneHot encoding technique, which uses dummy variables to encode categorical features (Liu, 2017). while the numerical variables were normalized with the Min-Max technique with a minimum value of 0 and a maximum value of 1. The response variable was encoded with the label encoding technique. Additionally, a characteristic reduction was performed for both datasets with the Chi-square statistical test, which measures the independence of the variables, eliminating the independent or irrelevant variables for the classification (Lantz, 2015).

For the *Dataset1*, the variables determined as dependent, after applying the Chi-square technique, were the datapath id, the input port and the number of packets sent. In the case of *Dataset2*, after performing the chi-square test, the following variables were found to be relevant: bandwidths of path-1, path-2, and path-3, the datapath id and input port. It is important to emphasize that the characteristic reduction applies to all the variables of the dataset except for the response variables.

Finally, both datasets were balanced by applying the SMOTE over-sampling technique, in which the minority class is oversampled by introducing synthetic examples of the minority class (Chawla et al., 2002).

Training and testing partitions.

All models use the 10-fold cross-validation technique to evaluate their performance. This technique randomly divides the dataset into k disjoint subsets of equal size, where one subset is for testing and the rest of the subsets are for training. In each trial, a test set is classified with a learning algorithm, obtaining the value of a validation metric. The average of k

validation metrics for all trials is the estimated percentage of a metric for the classifier (García López et al., 2006).

Models configuration.

All machine learning models use an empirical parameter configuration for both experimental datasets. Therefore, the ANN model uses a single hidden layer with 1000 neurons and a ReLU activation function used in modern neural networks as default recomendation (Goodfellow et al., 2016). The input and output layers use a ReLU and SoftMax activation functions, respectively. Also, this model was trained with 100 epochs and an Adam optimization function with a learning rate of 0.001, in addition, it uses a categorical cross entropy as loss function. The SVM uses a Polynomial kernel, with a cost coefficient $C = 10$, $\gamma = 1$ and a of degree 3. Finally, the kNN model was set with $k = 5$ and the Euclidean distance.

Assessment metrics.

We computed the area under the receiver operating characteristic curve (AUC) metric to validate the performance of the MLCs. The AUC metric indicates how capable the model is in distinguishing between classes. The ACC, precision, and recall metrics were also calculated to support the discussion of obtained results.

Selection criteria.

Since we explored three MLCs on each dataset, we selected the best model according to the following rule: the model with the highest AUC score is selected, in the event of a tie in the AUC score, the model with the highest accuracy is selected.

The implementation of all the classifiers was done in the Python 3.8.5 language with the scikit-learn (SKlearn) and Keras libraries (Chollet, 2015; Guido & L, 2009; Pedregosa et al., 2011). In the same way, the network application in charge of steering the traffic in the network was developed for the Ryu SDN controller (Kubo et al., 2014).

RESULTS AND DISCUSSION

According to the experimental setup section, we used three different MLCs with the 10-fold cross-validation method on both experimental datasets. The obtained results in terms of computed metrics highlighted successful performance for *Dataset1* and *Dataset2*, as could be read in Table *1*and Table *2*, respectively.

Method	AUC	Accuracy	Precision	Recall
ANN	0.9822	0.8592	0.8869	0.8556
SVM	**0.9994**	**0.9523**	**0.9584**	**0.9451**
KNN	0.9989	0.9964	0.9964	0.9967

Table 1: Metrics for ML Techniques for *Dataset1*

In Table *1* it can be seen that all the models reached a high AUC value, which indicates that they were able to distinguish between the 3 classes of the data set. In this case, the model that achieved the highest AUC was the Support Vector Machine, with a value of 0.9994 for AUC and 0.9523 for accuracy, however, a winning model cannot be defined because there are no significant differences between the 3 models. Empirically, the SVM performed slightly better in comparison with the remaining models.

In Figure *3* below, it shows the validation curves for the SVM model, these curves shows that the model is not over-fitting or under-fitting. The curves are displayed as a function of the gamma value.

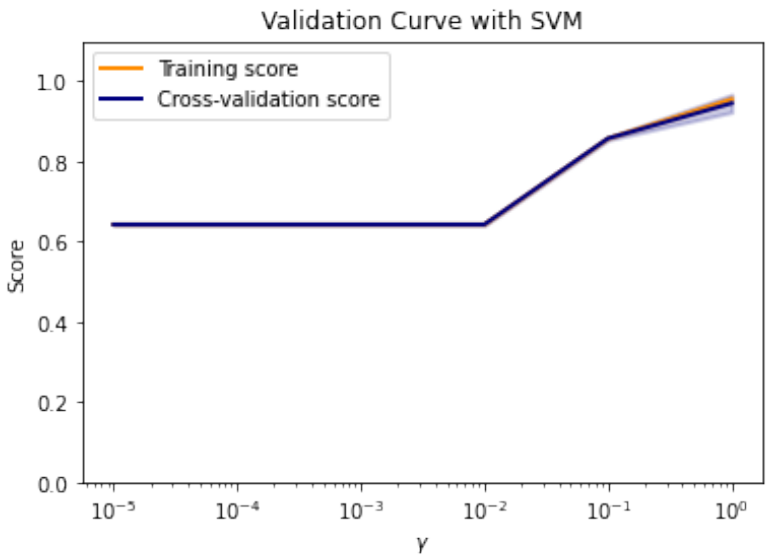

Figure 3: Validation curves for the SVM model by value of gamma *Dataset1*

Figure *4* shows the ROC and Precision-Recall curves, these curves indicate that good results were obtained for this model. The area under the ROC curve specifies that the SVM model was able to distinguish between each of the 3 classes, while a high value of the area under the Precision-Recall curve indicates a low rate of false positives and negatives.

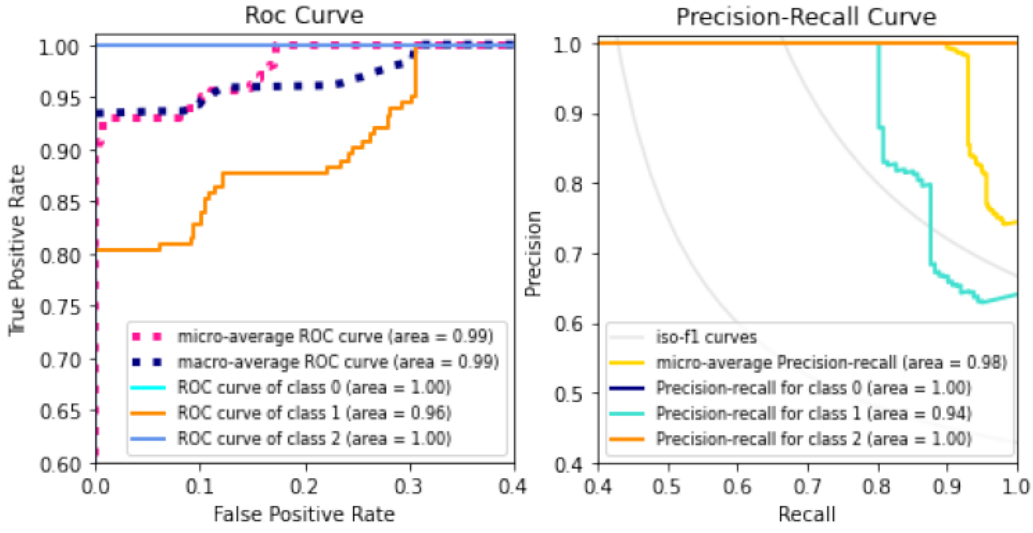

Figure 4: ROC and Precision-Recall curves for SVM *Dataset1*

Although the results regardinf the classification were encoraging, neither the SVM nor the other models achieved the desired alternation behavior on the paths of the network. The

models are limited to sending and receiving packets using a single route in the network. This means that the Dataset1 is not the right one to obtain the desired behavior.

Regarding the second dataset, all the models obtained an exceptionally good performance. The AUC metrics achieved are excellent, which indicates that each of the proposed models was able to correctly distinguish between each of the 3 classes. In addition the accuracy obtained is also particularly good, as can be read in Table *2*.

Method	AUC	Accuracy	Precision	Recall
ANN	0.9996	0.9877	0.9877	0.9877
SVM	**0.9996**	**0.9909**	**0.9909**	**0.9909**
KNN	0.9980	0.9844	0.9844	0.9844

Table 2: Metrics for ML Techniques for *Dataset2*

In the same way, the results shown in Table *2* do not allow designating a model as the best because there are no significant differences in the metrics of the 3 models. Empirically, the SVM model performed a little better in comparison with the remaining, due to its AUC and accuracy.

The validation curves for the SVM model are shown below in Figure *5*, where it can be seen that the model does not have over-fitting or under-fitting.

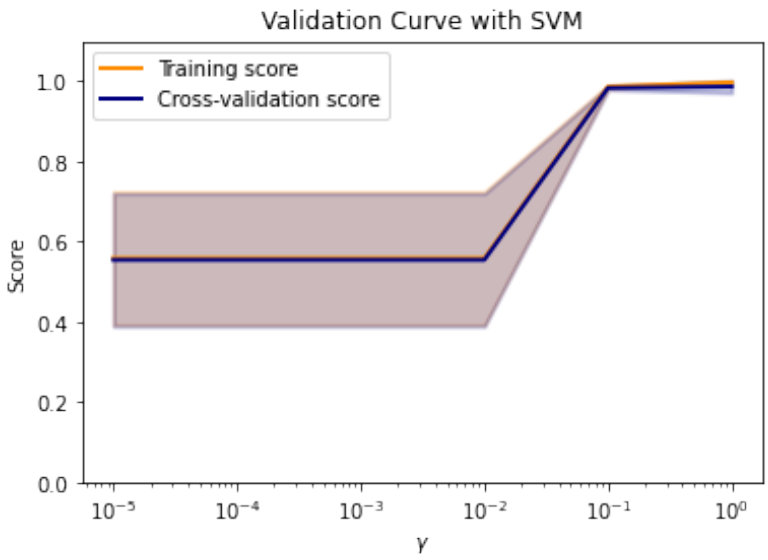

Figure 5: Validation curves for the SVM model by value of gamma *Dataset2*

Figure *6* shows the ROC and Precision-Recall curves, indicating that successful results were obtained for the SVM model and that the classification was particularly good, with a very low rate of false negatives and positives and a good distinction between classes.

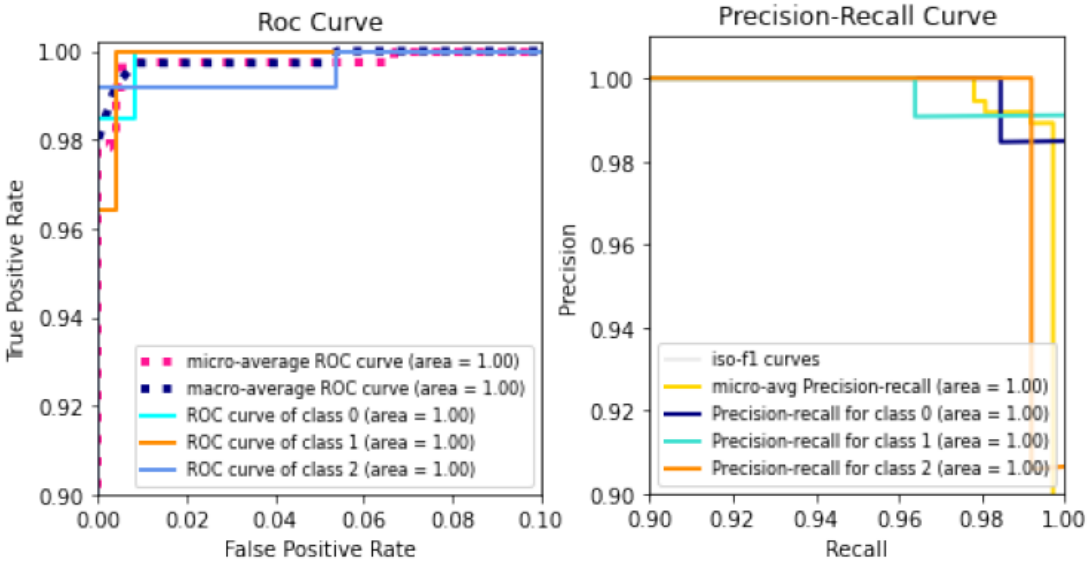

Figure 6: ROC and Precision-Recall curves for SVM *Dataset2*

In addition, it is important to mention that the SVM and the other models used in *Dataset2* were able of correctly direct the traffic for the proposed network, obtaining the desired behavior. These models provide alternation in the routes to send and receive packets,

when the last speed is greater than 5 Mbps, the models choose the next route with the lower bandwidth.

Finally, it is worth mentioning that for both datasets, all the models achieved an excellent performance, showing that the three have great generalization power. In addition, the data collected for each dataset helped the models to obtain excellent results in classifying the paths.

CONCLUSIONS AND FUTURE WORK

This work uses a software-defined network to obtain two datasets in order to feed 3 machine learning models and perform a network path classification. Specifically, the following classifiers were used: SVM, ANN and kNN, of which all managed to obtain good results. Despite the good results of all the models, no model could be designated as the winner, because there were no significant differences in their performance. Regardless of having excellent results for both datasets, in the first dataset, none of the classifiers were able to generate alternation in the network paths, which indicates that the dataset was not adequate for this behavior.

In the case of the second dataset, the algorithms did achieve the goal of creating alternation between the paths of the network, exchanging paths when the bandwidth of the current path was greater than 5 Mbps.

As future work, it is planned to use a network with a greater number of nodes such as the Abilene network, which have 11 nodes and 14 paths between the New York and Seattle nodes. This network would allow testing the performance of the machine learning models in a more complex network, as well as, to use reinforcement learning models to address the problem of classifying traffic on the network.

REFERENCES

Azodolmolky, S. (2013). *Software Defined Networking with OpenFlow*. Birmingham, UK: Packt Publishing.

Cai, Y., Qin, Z., Cui, F., Li, G. Y., & McCann, J. A. (2018). Modulation and Multiple Access for 5G Networks. *IEEE Communications Surveys & Tutorials*, *20*(1), 629–646. https://doi.org/10.1109/comst.2017.2766698

Candela, M., Luconi, V., & Vecchio, A. (2020). Impact of the COVID-19 pandemic on the Internet latency: A large-scale study. *Computer Networks*, *182*, 107495. https://doi.org/10.1016/j.comnet.2020.107495

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, *16*, 321–357. https://doi.org/10.1613/jair.953

Chollet, F. (2015). Keras. *GitHub Repository*. Published. https://github.com/fchollet/keras

Davies, G. (2019). *Networking Fundamentals*. Zaltbommel, Netherlands: Van Haren Publishing.

Fadlullah, Z. M., Tang, F., Mao, B., Kato, N., Akashi, O., Inoue, T., & Mizutani, K. (2017). State-of-the-Art Deep Learning: Evolving Machine Intelligence Toward Tomorrow's Intelligent Network Traffic Control Systems. *IEEE Communications Surveys & Tutorials*, *19*(4), 2432–2455. https://doi.org/10.1109/comst.2017.2707140

Fernández, D., Somavilla, J., Mateos, V., Walid, O., Rodríguez, J., Martín, M. J., Monserrat, F. J., Ferrer, M., & Galán, F. (Eds.). (2012). *Virtual Networks over LinuX (VNX)*. https://www.rediris.es/anuncios/2012/poster-terena.pdf

Forecast & G. M. D. T. (2019). Cisco visual networking index: global mobile data traffic forecast update, 2017–2022. *Update, 2017*, 2022.

García López, F., García Torres, M., Melián Batista, B., Moreno Pérez, J. A., & Moreno-Vega, J. M. (2006). Solving feature subset selection problem by a Parallel Scatter Search. *European Journal of Operational Research*, *169*(2), 477–489. https://doi.org/10.1016/j.ejor.2004.08.010

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning (Adaptive Computation and Machine Learning series)* (Illustrated ed.). The MIT Press.

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, *29*(7), 1645–1660. https://doi.org/10.1016/j.future.2013.01.010

Guido, V. R., & L, D. F. (2009). *Python 3 Reference Manual: (Python Documentation Manual Part 2)*. Createspace.

Kubo, R., Fujita, T., Agawa, Y., & Suzuki, H. (2014). Ryu SDN Framework—Open-source SDN Platform Software. *Feature Articles: Technological Development for Network Virtualization*, *12*.

Lantz, B. (2015). *Machine Learning with R: Expert techniques for predictive modeling to solve all your data analysis problems, 2nd Edition* (2nd ed.). Birmingham, UK: Packt Publishing.

Liu, Y. (2017). *Python Machine Learning by Example*. Van Haren Publishing.

Michael, A, K. J., Valla, E., Neggatu, N. S., & Moore, A. W. (2017, September). *Network traffic classification via neural networks* (UCAM-CL-TR-912). University of Cambridge, Computer Laboratory. Retrieved from https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-912.pdf

Moore, A. W., & Zuev, D. (2005). Internet traffic classification using bayesian analysis techniques. *Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems - SIGMETRICS '05*. Published. https://doi.org/10.1145/1064212.1064220

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. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*.

Pfaff, B., Pettit, J., Koponen, T., Jackson, E., Zhou, A., Rajahalme, J., Gross, J., Wang, A., Stringer, J., & Shelar, P. (Eds.). (2015). *The Design and Implementation of Open vSwitch*. https://www.usenix.org/system/files/conference/nsdi15/nsdi15-paper-pfaff.pdf

Shafiq, M., Yu, X., Laghari, A. A., Yao, L., Karn, N. K., & Abdessamia, F. (2016). Network Traffic Classification techniques and comparative analysis using Machine Learning algorithms. *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*. Published. https://doi.org/10.1109/compcomm.2016.7925139

Shu, J. H., Jiang, J., & Sun, J. X. (2018). Network traffic classification based on deep learning. *Journal of Physics: Conference Series*, *1087*. Published. Retrieved from https://iopscience.iop.org/article/10.1088/1742-6596/1087/6/062021/meta

Silva, P. H., & Alves, N. (2014). IPERF tool: generation and evaluation of TCP and UDP data traffic. *Notas Técnicas*, *4*(2), 1–13. https://doi.org/10.7437/nt2236-7640/2014.02.003