Classification of VPN network traffic flow using time-related features

Emmanuel Ayeleso
Faculty of Engineering
Computer Science
University of Ottawa
Email: eayel037@uottawa.ca

Alex Gagnon
School of Computer Science
Carleton University
Email: asgagnon@hotmail.com

***Abstract*—We propose the use of machine learning algorithms to classify network traffic flow over the Internet using time-related features as a better approach to classify VPN internet traffic effectively. The Sklearn package for the Python language was used to train ten algorithms across datasets consisting of 15, 30, and 120 second time related features. Stratified cross validation and statistical analysis was used to create classification models for VPN versus non-VPN network traffic. Our approach was also focused on creating a re-usable code base to quickly and easily examine differences between Sklearn compliant classifiers. Our results suggest Random Forest as the best classifier. Random Forest records accuracy of 86% with 15 seconds dataset as the best threshold.**

I. Introduction

VPN (Virtual Private Network) is a service that keeps activities over the internet private and secured. It works by routing traffic on the Internet through a VPN-tunnel that encrypts data and hides a user's IP address. Such VPN-tunnels make it difficult for ISP providers, hackers, man-in-the-middle listeners, governments, and other actors to snoop or view details of the the individuals network activities. This implies that VPN is an effective security and safety tool towards secured access to home, office, and public internet networks that is accessed through means such as Wi-Fi, Hotspots, cellular networks, etc. However, VPN is seen as a threat by some that may want to monitor and censor the activities going over the Internet. For instance, some countries want to monitor activities of their citizens over the Internet, and tertiary institutions' authorities often times want to restrict students' activities over their network in order to shun certain forms of connectivity, such as streaming films.

As a result, VPN is gradually becoming an enemy to these organizations who want to monitor and censor their members' activities on the Internet. These bodies have often resorted to banning VPN technologies entirely in their countries as means to curb its use. The problem is that such an outright ban infringes on the privacy and rights of the citizens and businesses that want to use VPN for genuinely valid and sensitive transactions that may require secure tunnels.

We seek to identify the use of machine learning algorithms to classify network traffic flow over the internet using time-related features as a solution to this problem. This will result in several benefits. Firstly, machine learning algorithms can be trained to classify traffic over the internet as a VPN or non-VPN traffic based on the application being used. This would allow these restrictive organizations a means to permit certain essential services (e.g. banking), while still being able to censor undesirable traffic (e.g. streaming). Furthermore, these algorithms can be used for monitoring and censorship agenda of these regulatory authorities. Lastly, knowing the effectiveness of machine learning algorithms is useful to the end user, as they will understand that even though they may be able to encrypt the contents of their network activities, they can still be identified as users of VPN technologies, which could be hazardous to their freedoms or well-being.

Our approach is without infringement on the direct privacy and security of the citizens and businesses. Considering the time factor, the scope of our work shall be limited to classification of the internet traffic as either VPN of non-VPN.

We have focused on training ten classification models (Decision Tree, SVM, KNN, Gradient Boosting, AdaBooster, Random Forest, Naive Bayes, Dummy Rule Based, SGD, and Neural Networks) with the datasets created by Canadian Institute of Cybersecurity. In addition, we applied modern advancements in the field to the experiments which enhanced our results in comparison to the related existing research efforts. To the best of our knowledge, no existing related research work used up to ten algorithms in their experiment. We chose 10 algorithms in order to compare statistically, the performances of at least a model representing linear, tree-based, distance-based, rule-based and ensemble classifiers. Additionally, effort was placed in creating a code base that would allow for consistent testing of the algorithms in such a way as to allow easy modification of feature selection, feature engineering, normalization, and classifier selection.

II. Description of the problem domain

Recently, the research community has shifted focus towards the use of machine learning techniques to classify internet traffic effectively [2], [3], [4], [6]. This is seen as a better approach to the diminished effectiveness of port-based and the computational overheads of the deep packet inspection that were the traditional approaches[6]. The concept of machine learning techniques basically relies on statistical patterns underlying the internet traffic. That is, using traffic features to

train machine learning models for classification. Such features in our area of research are: distribution of flow duration, flow idle time, packet inter-arrival time, and packet lengths of various applications. These features have been asserted by the researchers as an effective yardstick to classify the internet traffic [3], [1], [7].

Two research works motivated our study [1], [7]. Gerard et. al. studied the effectiveness of flow-based time-related features to detect VPN traffic. They used Weka to implement C4.5, decision-tree and KNN algorithms to classify their generated data sets using 10 folds cross validation. Sikha et. al also used the same data sets generated by Gerard et. al. study to train six different machine learning classifier models (logistic regression, SVM, Naive Bayes, kNN, Random Forest and Gradient Boosting Tree). They compare the performances of these models and recommended the best performing model for VPN and non-VPN traffic classification. As a contribution to the body of knowledge, we featured ten classifier models. Five of these classifiers (SVM, Naive Bayes, kNN, Random Forest and Gradient Boosting Tree) were featured by Sikha et. al. and we added the additional five other classifiers (Decision Tree, AdaBooster, Dummy, SGD, and Neural Network). Our interest was to study the performances of these models and possibly determine the best VPN and non-VPN traffic classifier.

III. Model Selection

We used ten models: Decision Tree, SVM, KNN, Gradient Boosting, Random Forest, Decision Tree, AdaBooster, Dummy, SGD, and Neural Network to train the datasets. Our choice of these models is informed by our understanding of the current body of knowledge in order to examine the performances of these models and see any insight that can be learned.

IV. Experimental setup

We downloaded the datasets and converted them into Panda dataframe recognised by Sklearn. Thereafter, we used several Python packages (matlibplot, scipy, numpy, etc.) to perform exploratory data analysis through statistics and visualization. Subsection A-E reported our observations and actions taken. For example, creating boxplots and histograms of the distributions of each feature, and a feature correlation heatmap were examined to determine where potential enhancements could be gleaned, such as removal of highly correlated data and choosing adequate normalization techniques.

A. Datasets

We used generated real-world traffic bench-marked intrusion detection datasets created by the Canadian Institute of Cybersecurity to train the ten algorithms. The data sets have 24 attributes generated under the time frames: 15, 30, 60 and 120 seconds and the instances of 76,379, 14,670, 15,238 and 10,801 respectively. Our findings revealed that 60 seconds time frame dataset had differing attributes than the other three, and so was dropped as result of its incompatibility with others.

B. Data Cleansing

We performed Exploratory Data Analysis (EDA) and discovered several notable findings. First, a default value of "-1" denoted missing values. We considered simply imputing values as replacements, however we felt that even after deleting every row (instance) that contained any missing value, we were left with enough samples to use in classifier training. Table I displays the summary of the deleted instances before and after the data cleansing across the datasets.

C. Feature Selection

We explored the results of the three datasets in our experiment. We believe that our actions will give room for the comparison of the chosen algorithms' performance against multiple datasets beyond the original dataset downloaded. Feature selection actions that we carried out on the datasets were:

1) Original Dataset:These are the downloaded datasets from the Canadian Institute of Cybersecurity without any feature selection.
2) Highly Correlation Feature Elimination Dataset (HCFFD1): During the EDA, we plotted the features correlation of the datasets with Pandas .corr() method (see Fig. 1). We believed that the highly correlated features may not be contributing significantly to the performance of the algorithms, and would be especially detrimental to models that use probabilities based on Bayes equation, such as the Naive Bayes classifier (which assumes mutual exclusion). For example, features that represent 'active' and 'idle' are in fact inverses, as whenever a connection is active, it cannot also be idle, and vice versa.
3) Highly Correlation Feature Elimination Dataset (HCFFD2): Purposely, we created variations of HCFFD1 datasets and labeled them with postfix HCFFD2. As many models require large amounts of computation (i.e. for generating hyper-parameter optimizations, computing and updating weights in deep learning networks, etc.,) we would like to remove features that don't contribute significantly in order to speed up training time. In this case, introspection of the features during EDA indicated that the min and max feature variations were highly non-normal, and would likely create skew.
4) Principal Component Analysis (PCA): We performed dimentionality reduction on the datasets with Pandas method PCA.fit_transform. Principal Component Analysis is the process of converting correlated variables into a smaller set of unrelated ones. In the same order of the HCFFD datasets, the goal is to reduce dimensionality, in the hopes of speeding up training time and possible raising (or at least not lowering by a significant degree) the overall performance of the classifier.
5) RFE - many classifiers enable ranking analysis of features in order to remove those that reduce performance

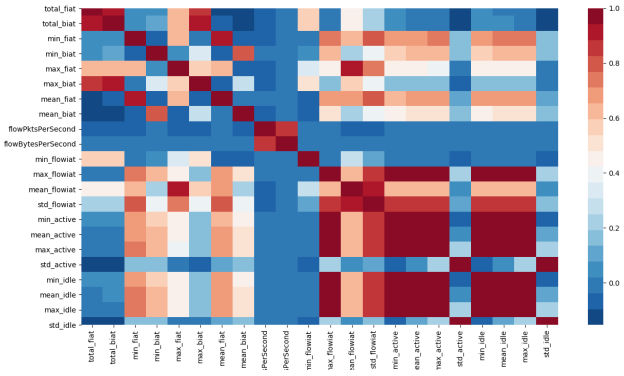

Fig. 1. Feature distribution histograms before normalization of the 15 second dataset

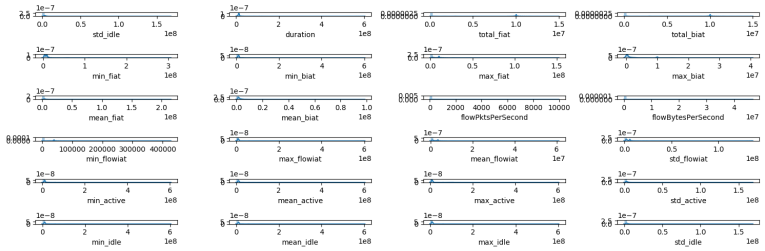

Fig. 2. Feature correlation heatmap used to generate manual feature selection methods

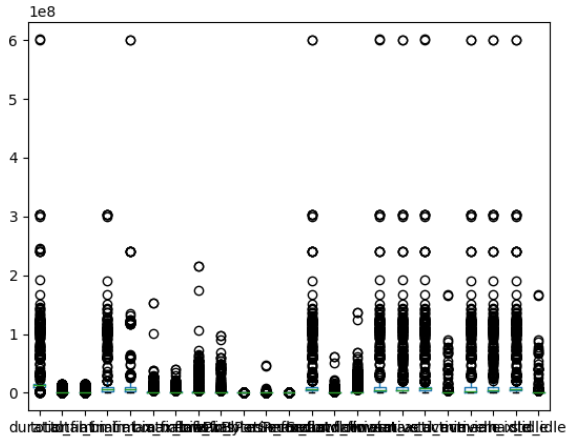

Fig. 3. Feature distribution boxplot before normalization of the 15 second dataset

or do not contribute significantly. Although it would be possible to create algorithms to compute rankings for those that do have have a native implementation (i.e. KNN, SVM), we did not have time to investigate such a solution. As this method of feature selection can be computationally expensive, and as one of our goals was to create a standardized way to evaluate various algorithms, this method was used only as additional insight and not used in further statistical analysis.

Each classifier therefore had performance metrics for each version of feature selection: none, PCA, two manual selections, and possibly RFE. Pairwise t-tests where used to examine if the results between the feature selection method were statistically significant. Interestingly, for most of the classifiers, the 'none' version (that is, no feature selection was done), proved to be the highest performing. This would be conducive to the original findings of the datasets creators.

In totality, we came up with four different datasets from each category of time-based datasets (15s, 30s and 120s). Such that 15s datasets produced 15s Original, 15s HCFFD1, 15s HCFFD2 and 15s PCA, and similarly for the 30s and 120s datasets. At the end of the feature selection exercise, a total of 12 different datasets were generated that we eventually used for training in the course of our experiments. This had the added benefit of allowing us to test significance of the algorithms' performances using the Friedman's Test.

TABLE I
SUMMARY OF THE DATASETS INSTANCES BEFORE AND AFTER DELETION OF THE MISSING FIELDS

Datasets	Instances before deletion	Instances after deletion
15 Second	18758	7095
60 Seconds	15515	7131
120 Seconds	10782	5159

D. Normalization

In the features of all of the datasets we observed non-normal distributions of values, which could lead to incorrect statistical inference and invalid performance results in certain classifiers (i.e. the presence of large numeric values dominate in situations where distances are used in classification, such as with KNearestNeighbours). We therefore normalized the dataset using Pandas dataframe function called normalizer.fit_transform. The actual normalizer used in reporting these results is the Normalizer class offered by sklearn, however in the spirit of creating code that is re-usable and modular, the code-base allows for easily swapping in other sklearn normalization/scaling types such as with a StandardScaler or RobustScaler. The Normalizer normalizes data around the unit norm (1). While better than extremely large differences, the unit norm may still pose challenges to distance-based learners due to the squaring of values. Fig. 2 and 3 show the feature distributions and boxplots of the dataset before normalization, and Fig. 4 and 5 show the results after normalization.

E. Model Training

We implemented the training of the ten (10) algorithms in sklearn. In order better measure the true performance of the classifier to the general population, a test-train split method was used to divide the samples into a set used to train the learner, and another set on which to test the classifier against. For each classifier against each datasets, the samples were tested using ten folds cross validation. We used stratification in order to subject the algorithms to the same sample set in

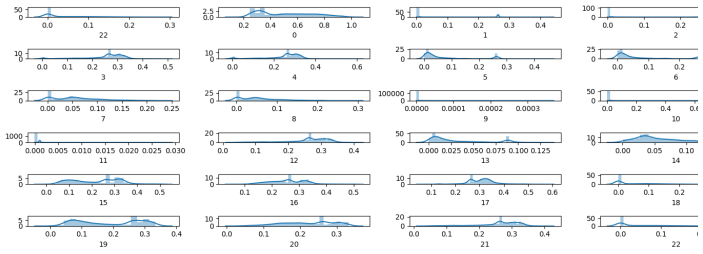

Fig. 4. Feature distribution histograms after normalization of the 15 second dataset

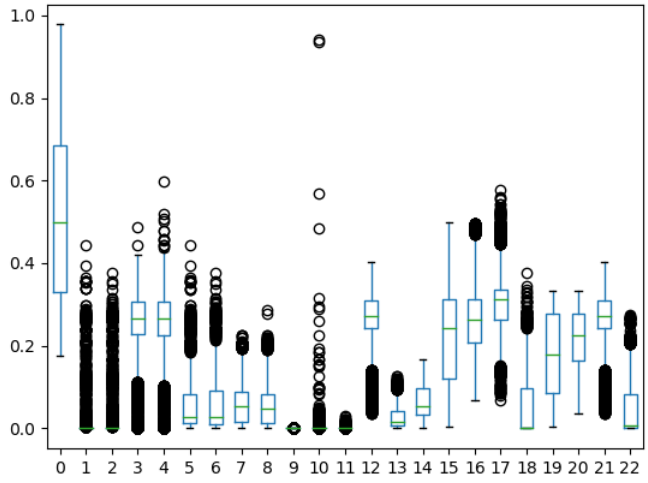

Fig. 5. Feature distributions after normalization of the 15 second dataset

each fold, in order to give more reproducible and comparable results. Table II presents the average accuracy of the models and their individual performance ranking of each fold. In each fold, the accuracy, precision, recall, and f1 measure were computed. Since the classes were very well balanced, the choice of accuracy as the scoring metric was deemed appropriate, but again in ensuring the code-base is highly modular, choosing any of these metrics is a very simple task.

V. Evaluation criteria

Beyond simply visually assessing the performance results of the models from Table II, we performed Friedman's test to know if the average rankings of the the algorithm display statistically significant differences. The null hypothesis states that all algorithms are equivalent (and so their ranks should be equal). The Friedman's statistic computed by our results is 100.98, with pvalue: 9.96e-18, at k=10, n=12, and alpha=0.05. Since this pvalue is lower than the value required for a confidence level of 95%, we will reject the null hypothesis that all algorithms perform equally. We then used Nemenyi's test to perform a post-hoc test. Nemenyi's test, displayed in Fig. 6, calculated the critical difference (CD) against which

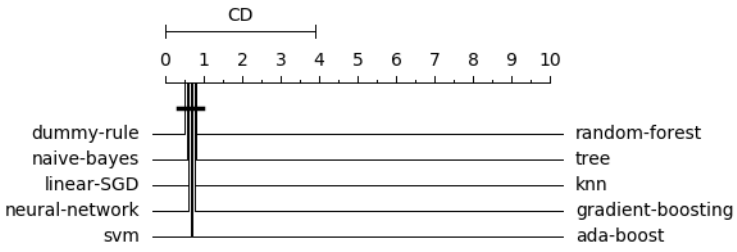

Fig. 6. Nemenyi Diadram showing the critical differences in the Performance of the algorithms against the datasets

the difference in the average rank between the algorithms are compared.

VI. Discussion of Results

Table II shows the the average accuracy of the models in order of the highest to the lowest accuracy as: Random Forest, Decision Tree, KNN, Gradient Boosting, Adaboost, SVM, Neural-Network, Linear-SGD, Naive Bayes, and Dummy. Friedman's test assert statistical significance in the performance of the algorithms. Fig. 6 visually presents the comparison of the classifiers. It reveals that Random Forest, followed by Decision tree significantly outperformed other classifiers. Random Forest classified 15s HCFFD1 dataset best with the overall mean accuracy of 86.33%, better than all other classifiers against eleven remaining datasets. The only exceptions are for the 15s PCA and 30s PCA KNN. It is interesting that for this domain, decision trees dominated above even other known well-performing ensemble learners such as AdaBoost and Gradient Boosting. This could be due to the lack of hyper-parameter tuning that is often required for these types of classifiers.

Our results suggest direct relationships between flow-timeout value and performance of the classifiers with 15 seconds as the best threshold. This could potentially be due to the extra encryption involved in VPN-tunneling. The 15 seconds threshold corroborate the findings of Arash et. al. (2016). Similarly, while the results of Sikha et. al (2017) suggest that Gradient boosting and Random Forest are the best classifiers of traffic flow-timeout, our results suggest the following order: Random Forest, Decision Tree, KNN and Gradient Boosting. It should be noted that Sikha et. al (2017) never considered Decision Tree in their experiment and that must have been the reason for its exclusion in their report. Our result also slightly contradicts the ordering of the performances of the classifiers as reported by Sikha et. al (2017). Our results show that KNN performs better that Gradient boosting as against the report of Sikha et. al (2017). Future work should include analysis of more classifiers such as those from different domains (e.g. deep learning, online learning), and inclusion of additional optimizations such as using GridSearch for hyper-parameter tuning.

TABLE II
PERFORMANCE ACCURACY AND RANKING OF THE TEN CLASSIER AGAINST 15S, 30S AND 120S DEATASETS USING 10 FOLDS CROSS VALIDATION

Datasets	Decision Tree	SVM	KNN	Gradient Boosting	AdaBooster	Random Forest	Naive Bayes	Dummy	SGD	Neural Network
15s Original	0.8134(2)	0.6670(6)	0.7502(4)	0.7787(3)	0.7243(5)	0.8485(1)	0.5819(9)	0.5180(10)	0.6221(7)	0.6062(8)
15s HCFFD1	0.8270(2)	0.6376(6)	0.7375(4)	0.7678(3)	0.7111(5)	0.8633(1)	0.5812(9)	0.5100(10)	0.6054(7)	0.5805(8)
15s HCFFD2	0.8101(2)	0.6464(6)	0.7495(3)	0.7451(4)	0.7065(5)	0.846(1)	0.5833(8)	0.513(10)	0.6182(7)	0.5795(9)
15s PCA	0.6908(3)	0.6580(6)	0.7334(1)	0.6802(4)	0.6617(5)	0.7088(2)	0.6101(7)	0.5150(10)	0.5736(9)	0.5789(8)
30s Original	0.832(2)	0.6329(6)	0.7557(4)	0.7870(3)	0.7395(5)	0.8563(1)	0.6297(7)	0.5046(10)	0.5970(9)	0.6181(8)
30s HCFFD1	0.8305(2)	0.6297(7)	0.7545(4)	0.7806(3)	0.7337(5)	0.8531(1)	0.6323(6)	0.5038(10)	0.5949(9)	0.6174(8)
30s HCFFD2	0.8126(2)	0.6210(6)	0.7627(4)	0.7707(3)	0.6869(5)	0.8337(1)	0.5830(9)	0.5164(10)	0.5916(8)	0.6194(7)
30s PCA	0.7243(3)	0.6617(6)	0.7509(1)	0.7076(4)	0.6675(5)	0.7481(2)	0.6166(7)	0.5164(10)	0.6012(9)	0.6120(8)
120 Original	0.7899(2)	0.6634(6)	0.7744(4)	0.7790(3)	0.7170(5)	0.8263(1)	0.4693(10)	0.4856(9)	0.5975(8)	0.5980(7)
120 HCFFD1	0.7926(2)	0.6603(6)	0.7720(3)	0.7647(4)	0.7121(5)	0.8080(1)	0.4628(10)	0.5137(9)	0.5997(7)	0.5985(8)
120 HCFFD2	0.7812(2)	0.6615(6)	0.7747(3)	0.7545(4)	0.6954(5)	0.7860(1)	0.4582(10)	0.5086(9)	0.6080(7)	0.5910(8)
120 PCA	0.7533(4)	0.6620(6)	0.77030(2)	0.8045(1)	0.6840(5)	0.7671(3)	0.5978(7)	0.4921(10)	0.5903(9)	0.5934(8)
Avg Rank	**2.3333**	**6.0833**	**3.0833**	**3.25**	**5**	**1.3333**	**8.25**	**9.75**	**8**	**7.9167**

VII. CONCLUSION

We trained ten algorithms with network traffic flow time related features to classify network as VPN and non-VPN. The datasets used consisted of 15, 30, and 120 second time-based features. With feature selection experimentation, we created a total of twelve datasets that were used to train the algorithms. Models produced in our experiments exhibit good predictive performance with Random Forest as the best with accuracy of 86%. Our recommendation for future work will be the use of more datasets instances to train the models used in this experiment. The outcomes of such experiments may give higher accuracy. Furthermore, algorithms used in this experiment could also be trained to classify network traffic according to the application that generated it (e.g. browsing, skype, streaming, etc.,) such that it will be possible to classify a network traffic based either VPN or not and the application that generated it.

ACKNOWLEDGMENT

We will like to thank the Canadian Institute of Cybersecurity for granting us free access to the datasets used during the course of our experiments.

REFERENCES

[1] Bagui, Sikha Fang, Xingang Kalaimannan, Ezhil Bagui, Subhash Sheehan, Joseph. (2017). Comparison of machine-learning algorithms for classification of VPN network traffic flow using time-related features. Journal of Cyber Security Technology. 1-19. 10.1080/23742917.2017.1321891.

[2] M. Shafiq, Xiangzhan Yu, A. A. Laghari, Lu Yao, N. K. Karn and F. Abdessamia, "Network Traffic Classification techniques and comparative analysis using Machine Learning algorithms," 2016 2nd IEEE International Conference on Computer and Communications (ICCC), Chengdu, 2016, pp. 2451-2455.

[3] T. T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," in IEEE Communications Surveys Tutorials, vol. 10, no. 4, pp. 56-76, Fourth Quarter 2008.

[4] Kim, Hyun-chul Fomenkov, Marina Claffy, Kc Brownlee, Nevil Barman, Dhiman Faloutsos, Michalis. (2009). Comparison of Internet Traffic Classification Tools.

[5] Manuel Crotti, Maurizio Dusi, Francesco Gringoli, andLuca Salgarelli.Traffic classification through simple statistical fingerprinting.SIGCOMM Comput. Commun.Rev., 37(1):5–16, January 2007

[6] Jeffrey Erman, Anirban Mahanti, Martin Arlitt, IraCohen, and Carey Williamson.Offline/realtime traffic classification using semi-supervised learning.Perform.Eval., 64(9-12):1194–1213, October 2007.

[7] Habibi Lashkari, Arash Draper Gil, Gerard Mamun, Mohammad Ghorbani, Ali. (2016). Characterization of Encrypted and VPN Traffic Using Time-Related Features. 10.5220/0005740704070414.

[8] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In KDD'01, pages 97–106, San Francisco, CA, 2001. ACM Press.

[9] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In KDD, pages 97–106, San Francisco, CA, 2001. ACM Press.