A Comparative Study on a Network Intrusion Detection Problem in an Online Setting and an Offline Setting

Zohreh Aghababaeyan
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Canada
zohreh.a@uottawa.ca

Ehsan Nazari
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Canada
ehsan.nazari@uottawa.ca

Yasaman Shahrasbi
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Canada
yasaman.shahrasbi@uottawa.ca

Amirhossein Zolfagharian
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Canada
a.zlf@uottawa.ca

***Abstract*—Detecting malicious activities from normal ones is an area of interest for many organizations, as it is of utmost importance for them to keep their digital privacy safe. Since it is not viable to completely protect systems from outside or inside attacks, making use of automatic systems such as machine learning techniques can play a crucial role in protecting sensitive systems. This paper tries to answer two main questions within a comparative study in an online and a novel offline ensemble machine learning setting. First, which of the online or offline machine learning settings would help systems better detect the attack behaviors, i.e., which setting has better overall performance, and second, which one of these two settings has fewer misclassified attack traffics than normal. Through this study and under a defined scenario, we discovered that the novel offline ensemble proposed in this paper outperforms the online learning model in terms of overall performance and the number of false-negative predictions.**

***Index Terms*—online learning, ensemble learning, network intrusion detection**

I. Introduction and Background

Using computer systems and the Internet is essential to profitability and sustainability of today's market due to the growing complexity of it. Ever increasing commercial services are provided via the Internet; The general direction of the human made systems is to enter the digital world. That in turn opens a whole new set of possibilities for security related issues and demands new set of tools to address digital vulnerabilities. Building digital systems which are robust against cyber-attacks are clustered into a domain called cyber-security. A major challenge in this field is detecting cyber-threats and handling them in time. Intrusion detection, the focus of this study, plays an important role in defending the networks against intrusions, attacks and malware. As a result, it is becoming increasingly important for organizations to invest in intrusion detection systems (IDS) to protect themselves against external threats and ensure network security.

- **Intrusion Detection:** Intrusion is any unauthorized activity from unknown attackers from outside of a system or insiders who try to exploit important and valuable data from computer systems. As the Internet is becoming an inseparable part of people's lives, the attackers seek new ways to misuse sensitive information and put the confidentiality, integrity, and availability of data and networks at risk.
  Intrusion detection systems are monitoring methods to detect malicious activities. This methods are meant to keep the systems and data safe. Two main groups of intrusion detection systems are **Signature-based IDS (SIDS)** and **Anomaly-based IDS (AIDS)**.

  – **Signature-based IDS (SIDS):** In Signature-based IDS, once a threat is revealed, it is added to the database of the threats. Incoming data are compared to the database of the detected threats, to detect whether it is malicious or not. SIDS can achieve a high performance with almost no false positives; the reason is that the predictions are based on the known malicious data. However the drawback of this method is the weakness of detecting zero-day vulnerabilities.
  From the SIDS point of view, network intrusion detection problem is formulated as follows: considering the stream of the data flowing in and/or out, we need to assign each data packet to one of the two classes representing normal traffic or intrusions

(where the intrusion could have different types, and thus it may also be a multi-class classification problem as well).

- **Anomaly-based IDS (AIDS)** As an other method in intrusion detection, Anomaly-based IDS makes use of machine learning or statistical-based methods to detect malicious data. As a result, it becomes more helpful to detect unseen data like zero-day attacks as opposed to the SIDS methods [12]; however, we may have more false positive predictions compared to SIDS.

A. *State-of-the-art Review*

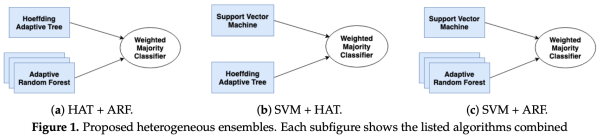
- **Ensemble**
  Ensemble methods combine several base learners (weak learners) rather than just using a single base learner to train a model with the hope of improving the overall performance. The justification behind the ensemble method is "The combined performance of many relatively uncorrelated models will be better than that of any individual model."
- **Online Setting**
  As an area in machine learning, online learning is where the data arrives in a sequence over time, i.e, all the training data is not available at the time of model training as opposed to the traditional batch learning. Two advantages of this setting is that in online learning, there is no need to store all the data, and the model can be updated as new data arrives without retraining the model from the beginning. [9]
  It is possible to categorize the architecture of online ensemble algorithms into different classes depending on what algorithms are used for base learners, how different single base learners are combined, how diverse they are, and what voting method is used to determine the final result. [9] Accuracy-weighted, weighted majority, stacking, bagging, boosting, and ensembles of Hoeffding trees are the main categories of online ensemble algorithms.
  In accuracy-weighted ensembles, each base learner is given a weight corresponding to the difference of error rates between that single base learner and a random classifier. One of the advantages of this algorithm is that non-streaming algorithms can be employed as base learners. [9]
  In weighted majority ensembles, a weight is assigned to each base learner. If a base learner predicts incorrectly, its weight reduces and increases otherwise.
  Stacking ensembles make use of a meta-learning architecture, the meta-learner (combiner) is trained on meta-data, which are the properties of the learning algorithm [18] or the outputs of base learners [2].
  In the batch setting, the base learners of the Bagging algorithm are given a bootstrap sample of the whole data set according to the binomial distribution; However, as the number of instances in the streaming setting is infinite, the sampling follows the Poisson(1) distribution, meaning that a weight according to Poisson(1) assigns to the arriving instance. Also, in the Online bagging algorithm, as the base learner any algorithm from sklearn package can be employed. [21]
  Online Boosting algorithms similar to the boosting algorithms in the batch setting build the models sequentially, meaning that larger weights will be assigned to the examples that are incorrectly classified so that the next algorithms will take those examples more into consideration. However, as opposed to the batch learning, where in most cases boosting algorithms outperform the bagging algorithms, in the streaming setting, this is not always the case especially as the boosting algorithms are sensitive to noise.
  The other type of online ensembles, the ensembles of Hoeffding trees are not able to outperform the proposed bagging and boosting algorithms; However, the Adaptive Random Forest algorithm have shown promising results. [3]
  - **Hoeffding Tree:** The Hoeffding tree algorithm is a very fast decision tree used for streaming data. It has been proved that the Hoeffding tree algorithm built through time in a streaming setting converges to a decision tree algorithm that is built by the whole dataset. Hoeffding tree algorithm is based on Hoeffding's bound; in essence, in each iteration, for splitting the data, the split gain, i.e., the impurity of the split, is calculated, and if the difference between the best attribute and the second-best attribute is greater than the Hoeffding bound, the tree will be split on the best attribute. [3] Info gain is used as the split criterion in this project.
  - **K-Nearest Neighbor.** In essence, knn which is a lazy learner is a batch model that can also be used naturally for incremental learning, and it is considered as a baseline classifier in online learning. In online learning, a sliding window will be searched by the knn model to find the class label of the incoming input instance. Moreover, it has been found that sliding windows with the size of 1000 instances is an optimal number. [3] The number of neighbors is chosen to be 3 for this setting.
  - **Adaptive Random Forest.** Random forest is bagging algorithm of the decision tree algorithms where it adds diversity to each classifier through sampling the data and selecting subsets of features for splitting the tree. Twenty hoeffding trees is used as estimators in this setting.

In Sheluhin et al [25], the goal is to compare four different algorithms in the online learning setting, i.e, Adaptive Random Forest (ARF), Hoeffding Adaptive Tree, K nearest neighbors, and Oza Bagging in a mobile application traffic classification problem. In this paper, for the window size, in the case of having balanced data, the

size of window expands as new data arrives; however, in the case of having imbalance data, the size of the window is fixed, and as new data arrives, the less useful instances will be dropped since the memory size is finite. The experiment shows that the ARF algorithm has the best result in terms of precision, recall, and it is almost three times faster than the other three algorithms.

In Lin et al [14], the authors design an adaptive online bagging ensemble for the problem of SDN intrusion detection in a streaming setting. Since the number of intrusions is more than normal instances, they employed a method which first balances the data, and then learns new, unseen data instances. In the "Online UnderOverBagging" algorithm, in the setting where data are imbalance, the rate of over sampling minority class increases and the rate of under sampling the majority class decreases over time. The novelty that is presented in this paper is that the probability of choosing majority samples and minority samples are not the same. The rate of choosing minorities is more than majorities; i.e, to choose majority samples, the Poission distribution is equal to 1 and to choose minority samples, the Poisson distribution is equal to an adaptive factor which is more than 1. In the experiment part, they used decision trees as their Online bagging base learner and compared it with AdaBoost and C4.5 algorithms. For the final result, the proposed method was able to improve the accuracy rate of detecting unknown intrusions for 0.056%.

In Martindale et al [16], authors compare several homogeneous, heterogeneous ensembles and stand-alone models addressing network intrusion detection in terms of performance and run-time expenses. Tested stand-alone models are K-Nearest Neighbours, Support Vector Machines(SVMs), and Hoeffding Adaptive Trees(HATs). For the homogenous ensembles, HATs and Adaptive Random Forests(ARF) are tested. Please refer to Fig. 1 for their proposed methods. This paper also presents



(a) HAT + ARF. (b) SVM + HAT. (c) SVM + ARF.

Figure 1. Proposed heterogeneous ensembles. Each subfigure shows the listed algorithms combined with a weighted majority classifier.

Fig. 1. Proposed methods of Martindale et al.

three novel online heterogeneous ensembles. The three approaches were ensembled together by using a weighted majority classifier. They tested all binary ensembles of the following solutions: HATs, ARFs, and SVMs. Specifically HATs + ARFs outperformed tested homogeneous ensembles. Overall, tested ensemble models resulted in better performance at the cost of higher runtime.

In Liu et al [15], authors focus on an ensemble of neural networks for online learning. They state that different random parameter initialization for neural network architecture might lead to different knowledge blind areas in the model. They suggest that ensemble learning could alleviate the mentioned problem. Their key idea is to initialize each model of the ensemble with random numbers drawn from different distributions. Please refer to Fig. 2 for their proposed architecture. They used 8 dif-

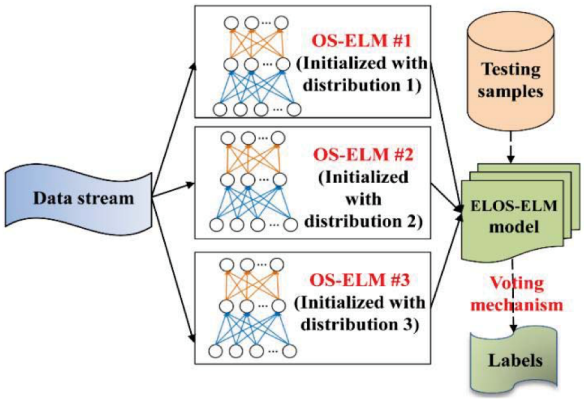


Fig. 2. Proposed methods of Lui et al.

ferent datasets to test their method against known Neural Networks with Random Weights. This idea effectively improves the diversity of the base models.

In Folino et al [8], authors shed light on strong points and weak points of neural networks in the context of online learning for intrusion detection. They mention that since all neural network classifiers have this assumption that the underlying distribution of the data that are being trained on is stationary, they cannot deal with the inherent feature of online learning in which the distribution of the data might change over time. They suggest that the non-stationary nature of the distribution of intrusion detection data can be properly addressed by using ensemble learning methods. They focus on homogeneous ensemble learning incorporating neural networks as their models. This raises the issue of even less number of labeled data for each neural network model of the ensemble. In order to address this issue they propose an incremental ensemble-based deep learning framework. Their novelty is to bring neural networks and chunk based learning techniques together. In Essence, their proposed framework combines several specialized neural network classifiers on different data chunks. The data chunks are outputs of a temporal segmentation of the input stream. Their framework also addresses the scarcity and unbalancedness of training data. They propose three different versions of ensemble methods, and compare the best of the three with the following six algorithms: OzaBagging algorithm: which is an adaptation of bagging algorithm, KNN, Hoeffding-Tree base learner, Online Boosting algorithm, Learn++.NSE. For each chunk of data in Learn++.NSE, one classifier is trained, and the classifiers are then combined based on a dynamically

weighted majority voting method. The weights are automatically determined by the accuracy of each classifier on the current and past data sets.. Their model outperforms the mentioned algorithms in terms of AUC score.
In Mirsky et al [19], authors propose a network intrusion detection system which is able to detect attacks in an unsupervised and online manner. They use an ensemble of neural networks to classify normal and abnormal traffic patterns. They use several autocoders each of which inputs the features of an instance and outputs root mean squared error(RMSE) of reconstruction of the instance; then all the RMSEs are input to another autoencoder(which could be thought of as a non-linear voting system for the ensemble); the final autoencoder calculates the RMSE and gives out the score for probability of intrusion. Their evaluations indicate that their model has a comparable performance in detecting intrusions to that of offline intrusion detection systems. Specifically, Kisune was compared to one Signature-based solution called Suricata, two Anomaly-based (batch) solutions called Iso. Forest and GMM, and two Anomaly-based (online) solutions called GMM Inc and pcStream. They tested their proposed algorithm on 9 different datasets and their AUC results were higher to equal to that of the other mentioned algorithms. Please refer to Fig. 3 for their proposed architecture. Also, their algorithm

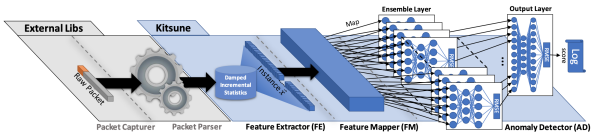


Fig. 3: An illustration of **Kitsune**'s Architecture.

Fig. 3. Proposed method of Mirsky et al.

is computationally so inexpensive that it can even be implemented on a Raspberry PI.

- **Offline Setting**
  The objective function approximation in offline learning systems does not change after training is completed.
  In an offline approach, all the data is ingested at once to build a model as opposed to an online approach where observation by observation is collected. [23]
  Learning online has the advantage of being data-efficient since the data is no longer required once it has been used, and we don't have to store it.
  - **Random Forest.** Random forest is an ensemble of several decision tree classifiers where each tree is trained on a bootstrap sample with replacement of the data. To split the data at each node, a subset of features are selected randomly instead of selecting all the features, which increases the diversity of the trees.
  - **Extremely Randomized Trees Classifier (Extra Trees Classifier).** Extra trees classifier is a variation of the random forest algorithm where all the trees are trained on the whole dataset instead of randomly choosing a subset of data per tree. Also, to split at each node, several features are randomly selected.
  - **Light Gradient Boosting Machine.** Gradient boosting algorithms consist of three main components, first, there is a loss function that is intended to be optimized in each step of the algorithm, which can be defined according to the problem at hand. Second, weak learners such as decision tree models are used as the base learners or weak learners in the boosting algorithm. Third, boosting algorithms work sequentially, i.e., each weak learner is added one at a time, and the gradient descent minimizes the loss function which in this case is that it tries to find the beset parameters for the tree such that it can be able to improve the performance of the model. [4] Moreover, light gradient boosting machine is an efficient open-source package to effectively implement the gradient boosting algorithm. [5]

Ensemble learning has gained a lot of attention in Cybersecurity domain, and the intrusion detection systems are not an exception. An ensemble learning so called committee based learning, is built upon several single (ideally non-correlated) classifiers, e.g. learners. Each learner is trained separately and predicts a class label, where the final prediction is made using a particular aggregating (voting technique) e.g. a majority voting.
Different methods of ensemble learning have been proposed. Some ensemble methods have similar learners but they focus on the Data to build uncorrelated (different) learners. In these methods, we create different data parts by bagging or boosting. As a second way, we can have learners Based on different algorithms or architectures trained on the same dataset, for example having a neural network model, a decision tree and one SVM classifier, where all of them are trained on the same dataset. Each of these learners predicts the label and based on voting we can extract a final label.
As the last method, we can have an ensemble method based on different features' scope. In the domain of network intrusion detection, essential features to detect an intrusion are extracted relying on expert knowledge about the characteristics of attacks. Previous papers on feature extraction for intrusion detection have proposed three main features set for network intrusion detection that are as follows [13] [7]:
Intrinsic features: general information related to the connection. e.g., duration, type, protocol, flag, etc.
Traffic feature: statistics representing similarity to previous connections
Content features: information about the content of the packets ("payload")
Each learner is trained in one of the mentioned feature scopes and the final predicted class is the result of voting between the results of each learners

- **UMAP**
  we rely on the Uniform Manifold Approximation and Projection (UMAP) [17] dimensionality reduction technique. We selected UMAP because several studies [6], [10] have shown its effectiveness as a pre-processing step to boost the performance of classification algorithms when compared to other state-of-the-art dimensionality reduction techniques such as PCA [11] and t-SNE [26]. In fact, PCA is a linear dimensional reduction technique that performs poorly on features with nonlinear relationships. Therefore, in order to deal with high-dimensionality data on low-dimensionality and nonlinear manifolds, some nonlinear dimensionality reduction algorithms such as UMAP and t-SNE should be used [10].
  However, t-SNE is more computationally expensive than UMAP and PCA. It is used in practice for data visualisation and data reduction to two or three dimensions. Furthermore, it involves hyperparameters that are not always easy to tune in order to get the best dimensionality reduction results.
  As a result, we relied in our study on UMAP for feature selection, as an effective pre-processing step to boost the performance of offline classification algorithms.

B. Problem Definition

In this project, we focus on proposing a novel offline ensemble learning method with feature selection and comparing it with an online-learning based model. These two models are compared based on Accuracy, F1-score, and kappa as the evaluation metrics and will also be compared in terms of execution times and the number of false negatives.

The main goal of this project is to compare the performance of an Online Bagging model and a new offline Ensemble method with combination of UMAP as a feature selection method in the context of intrusion detection problem. The dataset that we use in our project is NSL-KDD dataset [20]

NSL-KDD dataset contains records of internet traffic observed by a simple intrusion detection system.

For cyber-attack detection using artificial intelligence (AI), different machine learning (ML) algorithms have been suggested by many researchers, including Decision Trees, Random Forests (RF) and Support Vector Machines (SVMs) [1]

However, most of these proposed algorithms have a high false-positive rate, which leads to high costs and useless measures for the system and leads to the development of faulty intrusion detection systems. Another challenge in developing intrusion detection systems is the large number of data features that increases the processing time.

Therefore, in some articles, feature selection methods or reducing feature dimensions have been suggested. Recent papers published in these areas have proposed the use of the PCA [24] algorithm and transfer learning [22] to solve this problem effectively. This idea has had a significant impact on reducing processing time.

Therefore in this project, we aim to compare a new offline method by combining the UMAP algorithm for feature reduction with an online ensemble algorithm.

In summary, the key contributions of this work are as follows:

- Discarding redundant features using UMAP dimensionality reduction which improves intrusion detection models' computation time and detection rate.
- Soft voting ensemble by considering Extremely Randomized Trees Classifier (Extra Trees Classifier), Random Forest, and Light Gradient Boosting Machine effectively detects several types of malicious behaviours.
- Comparing a state-of-the-art Offline ensemble algorithm with an online ensemble algorithm in terms of evaluation metrics (Accuracy, f1-score, number of false negatives, and execution times.)

II. ALGORITHM/METHODOLOGY

We propose a scenario for comparing these two different approaches. Fig. 4, shows an overview of the proposed solution for intrusion detection. In the following paragraphs we explain the settings in depth.

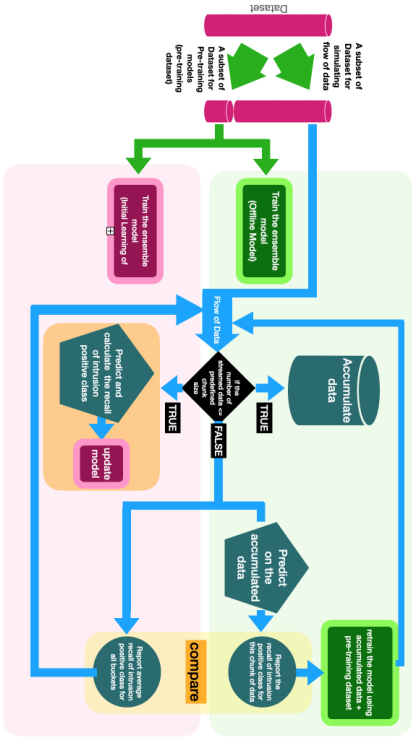


Fig. 4. The general overview of the proposed models.

This scenario is started by first setting aside a small portion of the dataset as a pre-training dataset. By using this pre-training dataset, we train our online and offline models. We

intend to compare the performance of the offline and online models over a specific period of time, for example, every 30 days. For this purpose, we divide the remaining dataset into N chunks. Suppose each chunk represents 30 days. (In other words thirty days is equivalent to seeing a chunk of data, for example, after seeing 1000 data, thirty days pass). Then during thirty days, we use the offline pre-trained model to predict the data; On the other side, according to the prequential evaluation procedure, using the streaming data, we first test the online model, and then incrementally update it. The online pre-trained model is updated after predicting a bucket of data and is tested on a new bucket in the next iteration(i.e. next day). After the end of thirty days, we compare the performance of the offline model to the performance of the online model, in terms of Precision, Recall, F1-score etc. Note that our dataset is divided to one pre-training sub-dataset and one other sub-dataset which is used to simulate the flow of data for online learning. This part consists of N chunks.

We also compare the two methods in terms of execution time. Then, to validate our results by using the remaining N-1 chunks of the data, we continue this 30-day interval as follows. At the beginning of the second thirty-day period, we train the offline model on all the data it has seen so far. ('all data' means the accumulation of pre-training dataset with the data it has seen in the past thirty days.) After that, we continue the process of predicting as before for the offline model, and we do not change the online model because the online model is updated after the end of each day.

It is noteworthy that the training process in the online and offline model is different (update in the online model and retraining in the offline model are not the same). It is not the case that the online model retains all the previous information. The second point is that the two online and offline methods used in this article are different, so deciding about which one of these two methods will perform better is not evident before experimenting.

As detecting attack traffics from normal traffics is a sensitive issue in many cases, our motivation is to understand which setting would enable us to detect attacks effectively and act accordingly.

The reason that ensemble methods are beneficial in online learning is that these methods can be used along with drift detection methods, addition, or removal of other base learners, which are the useful in the case of non-stationary or evolving data streams. [9]

III. Experimental Setup and Evaluation

A. *Dataset*

For the dataset, KDDTrain+.TXT and KDDTest+.TXT files from the Canadian Institute for Cybersecurity website are used. First, these two datasets are merged and shuffled. This dataset has 43 features, where the last one corresponds to the intensity level of the traffic, so this feature is removed as this project is intended to tackle a binary classification scenario. Moreover, to handle the three categorical features -protocol_type, service, and flag-, label encoder is fit and transformed on the whole dataset to convert them to numerical data. Next, the min-max scaler is fit and transformed on the dataset to normalize the data. Lastly, the UMAP feature reduction was applied to the dataset, and 35 features were obtained. After that, 20 percent of the data is used for pre-training the online and offline models, and the rest is used as the data stream.

B. *Experiments*

For the online models, Hoeffding tree, K-Nearest Neighbors, and Adaptive Random Forest algorithms with default values are used as base learners of the bagging online ensemble. To find the best models for the offline setting ensemble model, the dataset is splitted into 70% for training and 30% for testing, then almost 15 different models are trained on the training data, and the f1-score metric of the models are compared on the test data. The Random Forest, Extra trees classifier, and Light Gradient Boosting Machine are the models achieved highest score between all models Fig. 5.

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lightgbm	Light Gradient Boosting Machine	0.9873	0.9995	0.9837	0.9865	0.9851	0.9740	0.9740	0.4775
rf	Random Forest Classifier	0.9857	0.9992	0.9819	0.9846	0.9832	0.9707	0.9708	1.2842
et	Extra Trees Classifier	0.9854	0.9957	0.9827	0.9831	0.9829	0.9701	0.9701	1.4208
dt	Decision Tree Classifier	0.9826	0.9824	0.9769	0.9824	0.9797	0.9645	0.9645	0.1158
gbc	Gradient Boosting Classifier	0.9774	0.9985	0.9705	0.9765	0.9735	0.9538	0.9538	3.1733
knn	K Neighbors Classifier	0.9737	0.9925	0.9634	0.9748	0.9691	0.9462	0.9463	3.8967
ada	Ada Boost Classifier	0.9651	0.9952	0.9522	0.9659	0.9590	0.9287	0.9288	0.9342
lr	Logistic Regression	0.9466	0.9839	0.9288	0.9456	0.9371	0.8908	0.8909	2.5308
svm	SVM - Linear Kernel	0.9364	0.0000	0.9371	0.9185	0.9267	0.8705	0.8722	0.2950
lda	Linear Discriminant Analysis	0.9363	0.9797	0.9285	0.9233	0.9258	0.8700	0.8701	0.6092
ridge	Ridge Classifier	0.9359	0.0000	0.9273	0.9235	0.9253	0.8692	0.8693	0.0600
nb	Naive Bayes	0.7855	0.8167	0.9892	0.6688	0.7980	0.5869	0.6390	0.0525
qda	Quadratic Discriminant Analysis	0.7188	0.6923	0.5087	0.8428	0.5203	0.3867	0.4483	0.2525
dummy	Dummy Classifier	0.5719	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0375

Fig. 5. Results of applying 15 different algorithms on the training dataset to find the three best models for the offline ensemble model.

In this implementation, we have considered a window size of 5000. While the data is being accumulated in the window, the online model tests each incoming instance and updates the model. On the other hand, the offline model will wait until all the 5000 instances are accumulated, and then tests on all the new 5000 instances and update the model on the new 5000 instances as well as all the other data seen so far- all the data seen from the beginning. Then, the evaluation metrics, accuracy, f1-score, kappa, execution time, and the number of false negatives for both the settings are calculated after the window size number of data is observed and recorded. These steps is repeated until all the instances from the data stream is seen.

IV. Resuts and Discussion

In terms of execution time, offline model is almost 3.5 times faster than the online ensemble model. Moreover, As can be seen in Fig. 6, the accuracy of the offline model in all the windows are higher than the ones in the online model.

Also, the f1-score of both the attack class in the offline setting is higher than the online setting. Fig. 7, Fig. 8

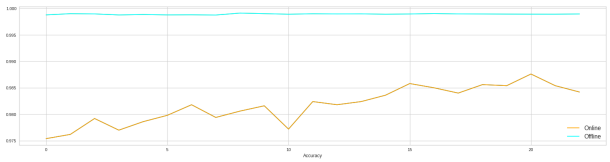


Fig. 6. accuracy of both online and offline settings

Fig. 7. f1 score of attack class in both online and offline settings

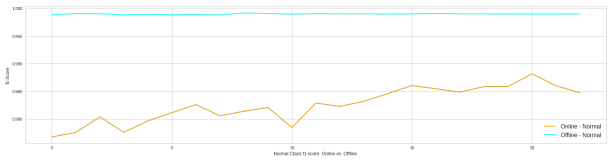


Fig. 8. f1 score of normal class in both online and offline settings

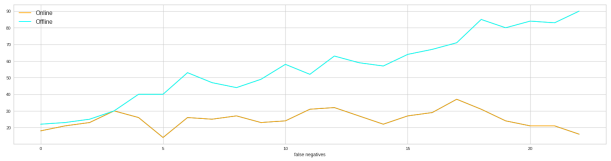


Fig. 9. false negatives of online and offline settings - window size = 5000

Looking at Fig. 9 and Fig. 10, it can be realized that changing the size of windows can affect the number of false negatives obtained.

The kappa results in Fig. 11 shows that in all the windows, the predictions of the offline model are very close to the true labels; However, the online model's predictions are very close to a random classifier.

V. Conclusion

Considering these results, we can conclude that under this scenario, the offline model has a better performance with respect to accuracy, f1-score, execution time, kappa, and the number of false negatives. Also, it is superior to the online setting in terms of time consumption, and it is easier to implement and has a straightforward theory. Moreover, as false negative predictions, i.e., misclassified attack traffic as normal, result in high costs in the real-world, it is important to take this value into consideration as well. In this implementation, we found that the offline model has fewer false negative predictions when having smaller window sizes. Interestingly on the other hand, when the size of windows grow, the online model outperforms the offline model.

VI. Lessons Learned

This work is helpful to understand the context of intrusion detection, and we found the proper ways to deal with the intrusion detection data. As a point to mention it was important that most of the oversampling methods caused a decrease in the accuracy of the models. By this project, we have checked the trade-off between online and offline models, and we observed the advantages and disadvantages of the models. However, it is truly accepted that more things need to be done to understand more aspects of this trade-off.

VII. Future Work

One of the **limitations of our model** is that having a great number of instances would increase the execution time, comparing the memory consumption of the two settings is one of the areas that we will focus as the future work. Also, as changing the distributions cannot be discovered by the offline model, using drift detection methods in the online setting can discover and tackle the problem of concept drift. Further investing the trade-offs between the window size and the number of false negatives in both online and offline settings and comparing different datasets in the intrusion detection domain are the areas that we will investigate as our next step.

References

[1] Bilgehan Arslan, Sedef Gunduz, and Seref Sagiroglu. A review on mobile threats and machine learning based detection approaches. In *2016 4th International Symposium on Digital Forensic and Security (ISDFS)*, pages 7–13. IEEE, 2016.

[2] Albert Bifet, Eibe Frank, Geoff Holmes, and Bernhard Pfahringer. Ensembles of restricted hoeffding trees. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(2):1–20, 2012.

[3] Albert Bifet, Ricard Gavaldà, Geoff Holmes, and Bernhard Pfahringer. *Machine learning for data streams: with practical examples in MOA*. MIT press, 2018.

[4] Jason Brownlee. A gentle introduction to the gradient boosting algorithm for machine learning, Aug 2020.

[5] Jason Brownlee. How to develop a light gradient boosted machine (lightgbm) ensemble, Apr 2021.

[6] Alex Diaz-Papkovich, Luke Anderson-Trocmé, and Simon Gravel. A review of umap in population genetics. *Journal of Human Genetics*, 66(1):85–91, 2021.

[7] Luca Didaci, Giorgio Giacinto, and Fabio Roli. Ensemble learning for intrusion detection in computer networks. In *Workshop Machine Learning Methods Applications, Siena, Italy*, 2002.

[8] Francesco Folino, Gianluigi Folino, Massimo Guarascio, Francesco Sergio Pisani, and Luigi Pontieri. On learning effective ensembles of deep neural networks for intrusion detection. *Information Fusion*, 72:48–69, 2021.

[9] Heitor Murilo Gomes, Jean Paul Barddal, Fabrício Enembreck, and Albert Bifet. A survey on ensemble learning for data stream classification. *ACM Computing Surveys (CSUR)*, 50(2):1–36, 2017.

[10] Yuta Hozumi, Rui Wang, Changchuan Yin, and Guo-Wei Wei. Umap-assisted k-means clustering of large-scale sars-cov-2 mutation datasets. *Computers in biology and medicine*, 131:104264, 2021.

[11] Ian T Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 2016.

[12] Ansam Khraisat, Iqbal Gondal, Peter Vamplew, and Joarder Kamruzzaman. Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2(1):1–22, 2019.

[13] Wenke Lee and Salvatore J Stolfo. A framework for constructing features and models for intrusion detection systems. *ACM transactions on Information and system security (TiSSEC)*, 3(4):227–261, 2000.

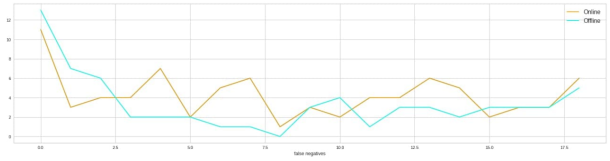


Fig. 10. false negatives of online and offline settings - window size = 500

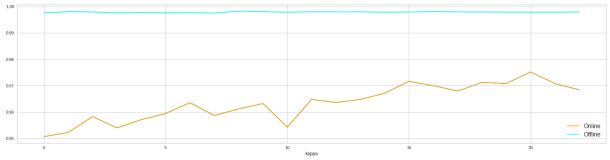


Fig. 11. kappa metric of both online and offline settings

[14] Zhang Lin and Du Hongle. Research on sdn intrusion detection based on online ensemble learning algorithm. In *2020 International Conference on Networking and Network Applications (NaNA)*, pages 114–118. IEEE, 2020.

[15] Ye Liu, Weipeng Cao, Yiwen Liu, and Weidong Zou. A novel ensemble learning method for online learning scenarios. In *2021 IEEE 4th International Conference on Electronics Technology (ICET)*, pages 1137–1140. IEEE, 2021.

[16] Nathan Martindale, Muhammad Ismail, and Douglas A Talbert. Ensemble-based online machine learning algorithms for network intrusion detection systems using streaming data. *Information*, 11(6):315, 2020.

[17] Leland McInnes, John Healy, and James Melville. Umap: uniform manifold approximation and projection for dimension reduction. 2020.

[18] Leandro L Minku and Xin Yao. Ddd: A new ensemble approach for dealing with concept drift. *IEEE transactions on knowledge and data engineering*, 24(4):619–633, 2011.

[19] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai. Kitsune: an ensemble of autoencoders for online network intrusion detection. *arXiv preprint arXiv:1802.09089*, 2018.

[20] Ghulam Mohi-ud din. Nsl-kdd, 2018.

[21] Jacob Montiel, Jesse Read, Albert Bifet, and Talel Abdessalem. Scikit-multiflow: A multi-output streaming framework. *The Journal of Machine Learning Research*, 19(1):2915–2914, 2018.

[22] Hassan Musafer, Abdelshakour Abuzneid, Miad Faezipour, and Ausif Mahmood. An enhanced design of sparse autoencoder for latent features extraction based on trigonometric simplexes for network intrusion detection systems. *Electronics*, 9(2):259, 2020.

[23] David Saad. Online algorithms and stochastic approximations. *Online Learning*, 5:6–3, 1998.

[24] Fadi Salo, Ali Bou Nassif, and Aleksander Essex. Dimensionality reduction with ig-pca and ensemble classifier for network intrusion detection. *Computer Networks*, 148:164–175, 2019.

[25] Oleg I Sheluhin, Viacheslav V Barkov, and Sergey A Sekretarev. The online classification of the mobile applications traffic using data mining techniques. *T-Comm-* , 13(10):60–67, 2019.

[26] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.