UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

Machine Learning Classifiers for IDS Construction

Alessandro David Viteri Martínez

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, 11 de enero de 2022

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

HOJA DE CALIFICACIÓN
DE TRABAJO DE FIN DE CARRERA

Machine Learning Classifiers for IDS Construction

Alessandro David Viteri Martínez

Nombre del profesor, Título académico **Daniel Riofrío, PhD.**

Quito, 11 de enero de 2022

© DERECHOS DE AUTOR

Nombres y apellidos: Alessandro David Viteri Martínez

Código: 137629

Cédula de identidad: 1725726671

Lugar y fecha: Quito, 11 de enero de 2022

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UNPUBLISHED DOCUMENT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

Currently, people tend to use a lot of mobile and web applications for their daily activities. It has become very common to make online purchases, to create many accounts (while sharing personal information) for different apps, and also to make money transfers thanks to the internet. For this reason, in the same way that it is important to have alarms to control who can enter to our home, it is also important to protect our networks and information by using Intrusion Detection Systems (IDS), in order to control non authorized entries or activity. For this paper it has been made a comparison between three of the most important and used Machine Learning Techniques for classification of data: Gaussian Naïve Bayes, Decision Tree and Support Vector Machine. Another interesting factor for this work was the use of two different datasets: Intrusion Detection and KDD-99. So the main objective was to find three best models for each dataset, it means, one for each ML classifier. So, in this paper there is a complete explanation about how the data was processed and how the models were created and trained since the variation of the parameters of each classifier. In general, we could say that the best classifier (based on metrics like precision, recall, f1-score and accuracy) was the DT, this algorithm obtained the best models for both Intrusion Detection and KDD-99 datasets. This algorithm performed an average of 0.99820 of all the metrics while classifying the data of KDD-99, and an average of 0.99797 with Intrusion Detection dataset.

Palabras clave: Machine Learning Classifiers - Intrusion Detection System (IDS) - Network Security - Naïve Bayes - Decision Tree - Support Vector Machine

ABSTRACT

Actualmente, las personas tienden a utilizar muchas aplicaciones móviles y web para sus actividades diarias. Se ha vuelto muy común hacer compras en línea, crear varias cuentas (mientras se comparte información personal) para diferentes aplicaciones y también hacer transferencias de dinero gracias a Internet. Por ello, así como es importante contar con alarmas para controlar quién puede ingresar a nuestro hogar, también es importante proteger nuestras redes e información mediante el uso de Sistemas de Detección de Intrusos (SDI), para controlar las entradas o actividades no autorizadas. Para este trabajo se realizó una comparación entre tres de las Técnicas de Aprendizaje Automático más importantes y utilizadas para la clasificación de datos: Gaussian Naïve Bayes, Decision Tree y Support Vector Machine. Otro factor interesante para este trabajo fue el uso de dos bases de datos diferentes: Intrusion Detection y KDD-99. El objetivo principal fue encontrar los tres mejores modelos para cada conjunto de datos, es decir, uno por cada clasificador de Aprendizaje Automático. Entonces, en este trabajo se explica completamente cómo se procesaron los datos y cómo se crearon y entrenaron los modelos a partir de la variación de los parámetros de cada clasificador. En general, podemos decir que el mejor clasificador (basado en métricas como precision, recall, f1-score y accuracy) fue el DT, este algoritmo obtuvo los mejores modelos tanto para la Intrusion Detection como para la base de datos KDD-99. Este algoritmo realizó un promedio de 0,99820 en todas las métricas al clasificar los datos de KDD-99 y un promedio de 0,99797 con la base de datos Intrusion Detection

Key words: Clasificadores de Aprendizaje Automático - Sistema de Detección de Intrusos (SDI) - Seguridad de Redes - Naïve Bayes - Árbol de Decisión - Máquina de Vectores de Soporte

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

In the same way that Internet and digital technologies are constantly developing since many years ago, the risks of keeping digital information secure rise too (Sharafaldin, Habibi Lashkari, & Ghorbani, 2018). People, currently, prefer to do the most of the daily activities like file sharing, providing personal information, searching information, mobile payment, and even, sharing our location by electronic devices. That is one of the major reasons why network security has become really important and more complex (Lee & Park, 2019). In order to protect personal information as a consequence of the increase of hacking incidents, the need to implement Intrusion Detection System (IDS) was created.

Therefore, an IDS is a security software/hardware system that alerts administrators when suspicious activity is discovered or identified in networks and computational systems, these IDSs can stop intrusions and block the potential threats. There is not only one type of IDSs; in fact, they can be classified in three groups: network-based, host-based and hybrid. A Network-based IDS monitors the network for suspicious traffic and is deployed in a strategic point or multiple points in the network. Moreover, a Host-based IDS analyzes and monitors inside of computers such as operating system audit records, application logs, and key system files for suspicious activity. And, an IDS could be hybrid when it can detect attacks from host and network sources (Atefinia, 2020).

An intrusion or attack can be accomplished in a system by exploiting the physical structure of the network for instance, hampering the network nodes, exploring the vulnerabilities of protocols used for communication or by injecting malicious code. Intrusions can be classified in four groups: physical intrusions, network intrusions, software intrusions and encryption intrusions. Physical Intrusions are concerned with hardware devices and affect the services and the processes performed on the devices and are capable of altering the data present in the

system resources. Network intrusions are related to routing of network packets, and can affect one or more nodes deployed in the network, also, these intrusions aim at harming the IoT network by disrupting the network activities or stealing the network information. Software intrusions tamper the data present in the system resources and even affect processing running in the system. The system intrusions includes malicious software programs in the form of virus, worms, malware, or Trojan horse. Additionally, encryption intrusions are concerned with compromising encryption process by tampering or stealing the public and private encryption keys (Thakkar, 2020).

Hence, the implication of Machine Learning techniques have become indispensable in network systems. Despite Deep Learning techniques are preferred than Machine Learning, because of their characteristic performance with large datasets, ML algorithms for classification can also get really good results, and in this paper their performance is going to be analysed. There are many types of ML classifiers, but for this work, we are going to analyse the performance and results of three algorithms: Naïve Bayes, Decision Tree and Support Vector Machine.

In this paper, two datasets are used, the first one is called Intrusion Detection (Kaggle, 2017) and can be obtained in Kaggle repositories. This one was used in used in A Comparison of Three Machine Learning Algorithms in the Classification of Network Intrusion (Zulhilmi, 2021), where the author compares three tree-based Machine Learning algorithms: Decision Tree (DT), Decision Jungle (DJ) and Decision Forest (DF). The results of this comparison said that the DF achieves the highest overall accuracy of 99.83% the DJ achieves the second highest overall accuracy of 99.74% and the DT achieves the lowest overall accuracy of 95.59%.

The second dataset is KDD-99 (Kaggle, 2018), although this dataset is more than 15 years old, it is still widely used in academic research. To investigate wide usage of this dataset in Machine Learning Research (MLR) and Intrusion Detection Systems (IDS) (Özgür & Erdem, 2016). For

example, in Machine Learning Algorithms In Context Of Intrusion Detection (Mehmood & Md Rais, 2016). In that paper a comparison between four ML algorithms was made: Support Vector Machine, Naïve Bayes, J.48 decision tree, and Decision Table for IDS construction in order to classify attacks. The results show that all the four algorithms had a good performance on classifying the data in the correct type of attack, and the overall comparison of different supervised algorithms says that SVM algorithm has an approximate 98% average of accuracy rate, the J.48 decision tree shows a better performance with an approximate 99% average of accuracy rate, NB classifier has the lowest average of accuracy rate with an approximate 85%, and Decision Table algorithm had an approximate 99% of average of accuracy rate.

DESARROLLO DEL TEMA

Materials and methods

For this paper, two datasets are used: Intrusion Detection and KDD-99. The most demanding tests were run in an NVidia DGX workstation of Universidad San Francisco de Quito. The source code was developed in Python 3.9. Also, three algorithms of classification were implemented: NB, SVM and DT. And, in order to find a good combination of hyperparameters, a GridSearch strategy was used.

Experimental databases.

For this research, two main datasets are going to be studied and used. Despite both datasets are related with Intrusion Detection Systems, each one has some special characteristics, for example, Intrusion Detection dataset manages more general groups of classification of attacks, and KDD-99 dataset manages more specific attacks. The first one is Intrusion Detection (Kaggle, 2017), the goal of this one is to learn a predictive model capable of distinguishing between legitimate and illegitimate connections in a network by presenting 42 features per entrance and a total of 125,973 entrances. Although, the most important features could be the protocol, information about the destiny and the origin of the connection and the “Type of Attack”. In this case, the “Type of Attack” is classified into four categories: DoS (Denial of Service), R2L (Root to Local), Probe, and U2R (User-to-Root). However, there is a fifth category that is “normal” and it marks that there is no attack on that object. Here, in Fig 1., there is a representation of the most important features of this dataset, the attacks. Taking into consideration that the number of entrances with attacks data does not have a good weight in comparison with the “normal” entrances, it is possible to say that this is a imbalanced dataset.

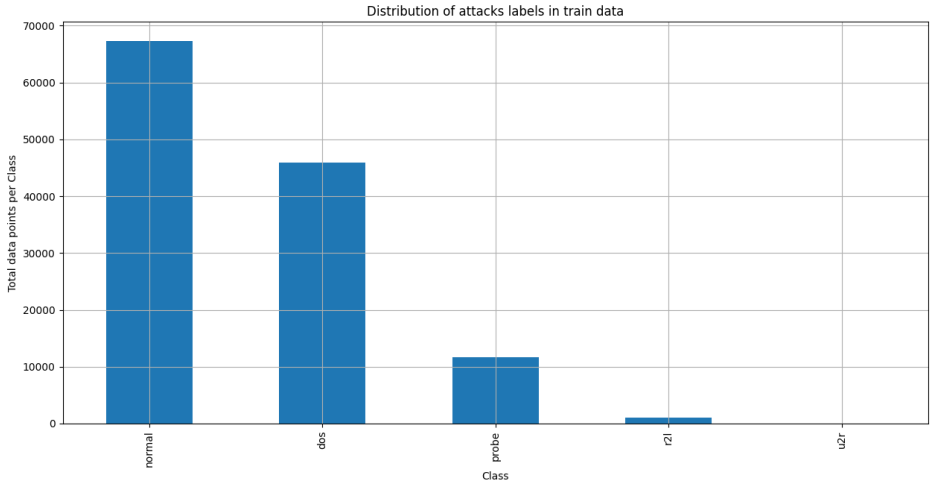

Figure 1. Data Structure of Intrusion Detection Dataset 1

The other dataset that was used is called KDDCUP'99 (Kaggle, 2018), which is widely used in academic research. This was the dataset used for The Third International Knowledge Discovery and Data Mining Tools Competition, which was held in conjunction with KDD-99 The Fifth International Conference on Knowledge Discovery and Data Mining. The competition task was to build a network intrusion detector, a predictive model capable of distinguishing between bad connections, called intrusions or attacks, and normal connections. This database contains 42 features of each entrance, and a total of 145,697 entrances; also, a standard set of data to be audited, which includes a wide variety of 22 intrusions like neptune, back, teardrop, etc. It also has a category "normal" when the object is not under attack, this data was simulated in a military network environment. In Fig. 2, there is a representation of the attack category, and it is an imbalanced dataset too.

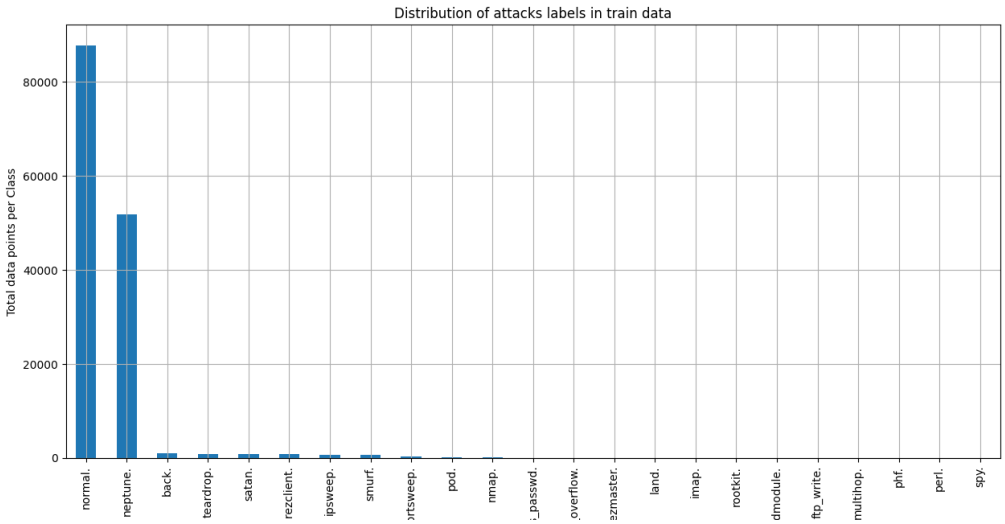

Figure 2. Data Structure of KDD-99 Dataset 2

Machine learning algorithms for classification.

Gaussian Naïve Bayes: The Naïve Bayes (NB) technique is a classification technique that computes the posterior probability of the given class using Bayes theorem. Thus, it evaluates the probability whether a particular feature set of given input sample categorizes into a specific class or not. For instance, for IDS, NB can classify the data samples of the network traffic as benign or anomalous based on the network traffic features. The traffic features are extracted from the network packet header and payload such as time duration of connection establishment, protocol used for communication, and connection flag status (Thakkar, 2020).There are some implementations for Naïve Bayes classifiers, in this work, we are using Gaussian Naïve Bayes (GNB), which is one of the simplest classification algorithms. It consists of assigning the label of the class that maximizes the posterior probability of each sample, under the assumption that the continuous values associated with each class are distributed according to a normal (or Gaussian) distribution, when working with continuous data. Therefore, it has a decision rule

that is based in terms of the discriminant function for each class at each intrusion classification (Ontivero-Ortega, Lage-Castellanos, Valente, Goebel, & Valdes-Sosa, 2017).

Decision Tree Classifier: Decision Trees (DT) are supervised machine learning methods used to classify data points according to attributes evaluated by a chosen metric. DTs are constructed from a set of instances following a divide and conquer strategy where if all instances belong to the same class, the tree collapses into a leaf with that specific class as label; otherwise, an attribute is selected to partition all data points according to the chosen metric or metrics (Quinlan, 1996). In Machine Learning applications, there are many fields where Trees can be put into practice, so, in this paper, we are using and analysing Decision Trees, using gini index for building the models. Gini index is a characteristic of Classification and Regression Trees (CART), these are used for classification or regression predictive working problem. Each node of the classification and regression tree model has two children only, which is a binary tree condition. One of the main advantage of Gini Index is that, it can distinguish any two distributions that has same entropy measure. This is because, any distribution cannot be summarized with a single measure (Daniya, Geetha, & Suresh Kumar, 2020).

Also, Decision trees are constructed recursively through an induction process with a top-down strategy, from general concepts to particular examples. This is why the acronym TDIDT ("Top-Down Induction on Decision Trees") is used to refer to the family of algorithms for constructing decision trees (Martínez Torres, Iglesias Comeseña, & García-Nieto, 2019).

Support Vector Machine: Support Vector Machines (SVM)s are set of related supervised learning methods used for classification and regression. They belong to a family of generalized linear classification. A special property of SVM is that it simultaneously minimizes the empirical classification error and maximizes the geometric margin. SVMs are also called Maximum Margin Classifiers. SVM is based on the Structural risk Minimization (SRM). SVM

map input vector to a higher dimensional space where a maximal separating hyperplane is constructed (Srivastava & Bhambhu, 2009). Traditional SVM can only deal with plain dataset and cannot tackle heterogeneous datasets directly, but with some refinement of kernel functions it can be extended on heterogeneous datasets (Kotpalliwar & Wajgi, 2015). Here RBF (Radial Bias Function) kernel is used for classification of heterogeneous data. RBF kernels are the most generalized form of kernelization and is one of the most widely used kernels due to its similarity to the Gaussian distribution. The RBF kernel function for two points computes the similarity or how close they are to each other. When the points are the same, there is no distance between them and therefore they are extremely similar, and when the points are separated by a large distance, then the kernel value is less than 1 and close to 0 which would mean that the points are dissimilar (Sreenivasa, 2020).

Experimental setup.

Once the datasets were pre-processed, the performance between three algorithms of classification is analysed. The algorithms that are used are NB, DT, and SVM. So, as we have two databases, there is going to be six "best models" to be analysed taking into consideration the performance of each algorithm over both datasets in the most important parameters for the classifying algorithms like confusion matrix, we will also calculate precision, recall and weighted f1-score to determine the best model of each model. In order to generate the best ML classifiers models, we trained 2280 different ML models that were based on a set of hyper-parameters in cross validation and five hundred of models for each algorithm, and then it had to be replied for both datasets. The source code was implemented in Python 3.9 using sklearn and pandas libraries.

Database preprocessing.

One of the first challenges was to find the similarities between the databases in order to know which features must be used for the analysis of the performance of the algorithms, it helped to make a fair comparison, because the same features are used in both datasets. Despite that the datasets are used in other investigation cases related to the same subject (IDS construction, usages, etc), they are built in different ways. For this reason, it was necessary to study the attributes of each dataset, and it was found that they actually share most of them. For example the protocol, service, flag, src-bytes, dst-bytes, land, attacks, etc. Only two them were different, those were not used for the study, and another difference was that service and flag were numerical attributes in dataset 1, and categorical in dataset 2, therefore, they had to be processed in a different way.

After, an important step was to clean the available data before using it for Data Analysis and building models. Some important steps involved in the data cleaning process were removing/inputting NULL values and removing duplicates from the dataset, as a technique to avoid unnecessary data that could affect the results (Singh, 2020). In Intrusion Detection dataset, NULL values were not found, but 32 duplicate entrances were deleted, so 125,941 entrances were used from this dataset. In KDD-99 dataset, NULL values were not found, but 111 duplicate entrances were deleted, so 145,586 entrances were used from this dataset.

Then, it was made an Exploratory Data Analysis as an approach for analyzing the dataset to summarize their main characteristics. This step was made in order to see that these datasets are highly imbalanced. Figure 1 and Figure 2 show the distribution of the data regarding each datapoint class.

Another important step was to vectorize categorical data using one-hot encoding. KDD-99 dataset has 3 categorical features: protocol, service and flag, these were vectorized using the

one-hot encoding method. For Intrusion Detection dataset, which only have the protocol as a categorical attribute, we apply the same method.

Finally, we made the step of standardizing the features. Data standardization is a data processing workflow, which main goal is to convert the structure of different datasets into one common format of data (Aggarwal & Kumar, 2021).

Training and testing.

For this section the sklearn method "train-test-split" was used in order to get a manual partition of the data. So, we divided randomly the entire datasets into two parts, where the training data is 75% of our total data and the test data consists of points belonging to the remaining 25% of the data. We use k-fold cross-validation (not the satisfied k-fold cross-validation) on the train data using Grid-search CV while building the models.

Hyperparameters.

For each algorithm we selected a list of hyperparameters to create models. Each algorithm has its own list of hyperparameters. Please refer to Table 1 to see the variation of these parameters.

Gaussian Naïve Bayes: the only hyper-parameter used for this algorithm is called varsmothing. It is usually defined as a portion of the largest variance of all features that is added to variances for calculation stability (Mayang Sari, Rahman Wijaya, Hidayat, & Kannan, 2021).

Decision Tree Classifier: the firs parameter is criterion, and it Refers to the attribute selection measure used to select attributes at each node in the DT. For this work, we are using Gini impurity. The second one is Max Depth, which indicates the higher expansion of the tree until the last branch node, also until leave node it's pure (Vaca, Riofrío, Pérez, & Benítez, 2020). The third one is Sample Leaf or Sample Split, it refers to minimum number of samples needed to split each internal mode (Vaca, Riofrío, Pérez, & Benítez, 2020). And the last one is the

Splitter, this is the strategy of DT training algorithm and chooses an attribute to split the dataset (Vaca, Riofrío, Pérez, & Benítez, 2020). We used the parameter “best” which is the exhaustive best split.

Support Vector Machine: the first parameter is the C value, the default value is 1, but as we have imbalanced and noisy data, it is going to take some lower values for more regularization.

The second one is the gamma value, which optimizes the fit of the model, The lower values of gamma result in models with lower accuracy and the same as the higher values of gamma (Ajitesh, 2020).

GridSearch: as it was explained before in this section, this is not an algorithm of classification, but it was made in order to make a “more fair” comparison between the three techniques used in this work. Said that, we used the same GridSearch parameters for every algorithm, these were a cross validation of 10 folds, a n-jobs of -1 in order to use all the computer's processors and a verbose of 1 for printing methods of the results.

Algorithm	Parameter	Range	Step Size
GNB	var-smoothing	(-10 to 3)	1
DT	max-depth	(3 to 30)	3
	min-samples-split	(3 to 15)	3
SVM	C value	(0.1 to 10)	1
	gamma	(1.1 to 0.01)	0.1

Table 1.Representation of the variation of hyperparameters per each algorithm

Selection criteria.

The best models are going to be selected by the best scores of their metrics Precision, Recall, Accuracy and F1-score. Also it is being calculated the average of all the metrics.

Metrics.

Metrics are a fundamental basis in the goal of measuring the best models and algorithms, for this research work, the metrics are presented in the following list:

- Precision: percentage of correct positives over the total number of positives identified (Martínez Torres, Iglesias Comeseña, & García-Nieto, 2019).
- Recall: percentage of positive items correctly identified as harmful over the total (Martínez Torres, Iglesias Comeseña, & García-Nieto, 2019).
- F1-score: this comes from precision and recall, this parameter measures the accuracy of the method (Martínez Torres, Iglesias Comeseña, & García-Nieto, 2019).
- Accuracy: percentage of correct predictions (positive and negative) (Martínez Torres, Iglesias Comeseña, & García-Nieto, 2019).

Results and discussion.

As it was planned and mentioned in the Experimental Setup section, a large number of models were created under the parameter proposed. Actually, 2280 models in total were trained and tested, this means that in average, about to 500 models were created for each dataset and each ML classifier. In order to give an overview of the results, two tables were created (see Table 1 and Table 2).

Seeing this, the first result could be that Decision Tree Classifier is the best algorithm, because it has excellent performances with both of the datasets by not having lower metrics scores (precision, recall, f-1 score and accuracy) results under 0.997. That is because the DT used in this paper is configured by the best parameters, also it is the fastest algorithm talking about of computational time. SVM is the second best algorithm, because it shows good results too. However, it has a disadvantage in comparison with DT. Referring to the time of processing that SVM needs to classify data. There is no doubt that this was a really good challenge for all

the algorithms, because for Intrusion Detection dataset we are classifying 5 categories of attacks, and for KDD-99 dataset, we are classifying 23 categories. But, the difference of the time of processing is too big and is too big. In the most demanding tests for DT and GNB techniques (using a cross-validation of 10 sets, and optimizing all the parameter), the time of processing did not pass one minute, on the other hand, when the most demanding tests were made to SVM, the time easily reached 4 to 5 hours.

Havig said that, we could place the GNB algorithm in third place, it is important to say that we are not saying that this is a bad ML technique for classification problems, of course, it has advantages, but it did not have a very good performance with dataset 1, where it has some metric that are under 0.85 of score. It does not have a be a completely bad characteristic, because this algorithm did an excellent job with dataset 2, where the metrics are not under 0.961 of score.This helps us to determinate that this is an algorithm that has a direct relation between the number of labels and its performance when it classifies, because dataset 2 has more characteristics to be classified than datatset 1, and this also shows that this algorithm works in a very efficient way when it works with large dataset, which not happens with other algorithms that lose information when they threat with big datasets.

There is another important aspect to analyse here, confusion matrices give us too a clear idea of how good the algorithms are classifying. A confusion matrix contains information about actual and predicted classifications done by a classification system. Figure 3 shows the confusion matrix for a two class classifier. Classification accuracy, sensitivity, specificity, positive predictive value and negative predictive value can be defined by using the elements of the confusion matrix (Fatih, 2009).

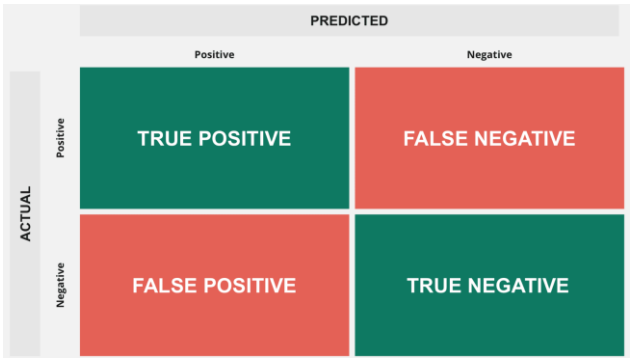

Figure 3. Structure of Confusion Matrix

In this way, it is important to understand that, despite the results are very good as for metric scores, this tool shows that this classification of data actually has some problems. This comes from the nature of the datasets, and is also analysed in Future Work section. So, confusion matrices show that a binary classification problem is appearing, this is because the datasets are highly imbalanced and when the algorithm does not know what to do, it classifies data in the two biggest classes that we have. Here is an example of that (see Figure 4). We can see there, that most of the data of those labels that do not have a good weight of entries, are being classified as "normal" and "dos" which is the attack with the best weight of this feature.

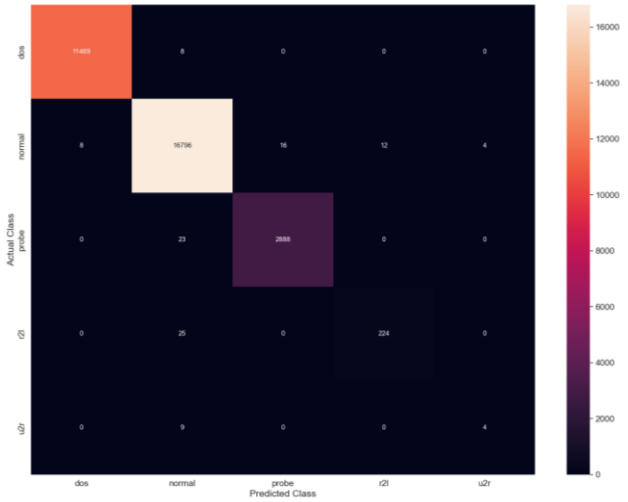

Figure 4. SVM Confusion Matrix for Intrusion Detection dataset

Database	Algorithm	Hyper-Parameters	Precision ± σ	Recall ± σ	F1-Score ± σ	Accuracy ± σ
Intrusion Detection	**GNB**	var-smoothing=1 cross-validation-sets=10	0.891 ± 0.076	0.884±0.081	0.886±0.080	0.884±0.081
	DT	max-depth=30 min-samples-split=3 cross-validation-sets=10	0.997±0.001	0.997±0.001	0.997±0.001	0.997±0.001
	SVM	C=100 gamma=0.1 cross-validation-sets=10	0.996±0.002	0.997±0.002	0.997±0.001	0.996±0.002

Table 2. Best 3 models for Intrusion Detection dataset

Database	Algorithm	Hyper-Parameters	Precision ± σ	Recall ± σ	F1-Score ± σ	Accuracy ± σ
Intrusion Detection	**GNB**	var-smoothing=1 cross-validation-sets=10	0.961 ± 0.027	0.972±0.019	0.965±0.024	0.972±0.019
	DT	max-depth=30 min-samples-split=3 cross-validation-sets=10	0.998±0.001	0.998±0.001	0.998±0.001	0.998±0.001
	SVM	C=100 gamma=0.1 cross-validation-sets=10	0.998±0.001	0.998±0.001	0.998±0.001	0.998±0.001

Table 3.Best 3 models for KDD-99dataset

CONCLUSIONES

As it was said in the discussion section about the results, these classifiers worked really well, each one by optimizing its parameters in order to find the best models. Despite it was done a very good job, there are a few things that could be improved in the future. We had to be very analytic for doing this, but as a part of an investigation work, there is always some aspects that could be improved. In this case, we can see that we had a really serious problem that came from the nature and behaviour of the datasets. The problem here was that both datasets are highly imbalanced, and in the case of dataset 2 (KDD-99) there are too many types of attacks that need to be classified, because it actually has 23 labels of them.

So, there is a somewhat serious problem behind all this, since in some fields there is very little information, which makes it very difficult for algorithms to predict accurately. For example, if we have datasets with approximately 130 thousand entries and only 2 of them belong to an attack such as “spy”, it is very unlikely that it will predict them well, since we have too little information about this event to classify it correctly, Therefore, for this parameter, you could have metrics with values of 0.5 or even 0.0. Therefore, it can be said that we are beginning to have a decline in the classification behavior and this is beginning to become a “binary classification” problem, which was not the main objective at first. But, this occurs, since the algorithms begin to classify the inputs with little information in two larger groups (two with the most information) which in this case would be “normal” that is not under attack and the class “neptune” which is the most numerically representative attack type in dataset 2, and class “dos” for dataset 1 (Intrusion Detection).

What is proposed, then, as future work, is to try to process the data from both databases, to convert the whole problem into one of binary classification, that is, all types of attacks could be converted into a single class attack and the rest of the data would belong to the class

“normal”. Undoubtedly, doing this would greatly improve the performance of the algorithms and their performance, since it would not have the need to classify based on badly made probabilities that are caused by the failure to classify many small classes into two majorities.

REFENCIAS BIBLIOGRÁFICAS

Aggarwal, S., & Kumar, N. (2021). Chapter Twenty-Two - Transportation system. In S. Aggarwal, N. Kumar, & R. Pethuru, *Advances in Computers* (pp. 431-454). Punjab: Thapar Institute of Engineering & Technology.

Ajitesh, K. (2020, 07 18). *SVM RBF Kernel Parameters with Code Examples*. Retrieved from Data Analytics: https://vitalflux.com/svm-rbf-kernel-parameters-code-sample/#:~:text=Gamma%20and%20C%20values%20are%20key%20hyperparameters%20that,values%20meaning%20%E2%80%98far%E2%80%99%20and%20high%20values%20meaning%20%E2%80%98close%E2%80%99

Atefinia, R. (2020). Network intrusion detection using multi-architectural modular deep neural network. *The Journal of Supercomputing*, 1-3.

Daniya, T., Geetha, M., & Suresh Kumar, K. (2020). CLASSIFICATION AND REGRESSION TREES WITH GINI INDEX. *Advances in Mathematics: Scientific Journal 9*, 3-6.

Fatih, M. (2009). Support vector machines combined with feature selection for breast cancer diagnosis. *ScienceDirect*, 4-5.

Kaggle. (2017, 10). Retrieved from Intrusion Detection: https://www.kaggle.com/what0919/intrusion-detection

Kaggle. (2018, 05). Retrieved from Network Anomaly Detection: https://www.kaggle.com/anushonkar/network-anamoly-detection

Kotpalliwar, M., & Wajgi, R. (2015). Classification of Attacks Using Support Vector (SVM) on KDDCUP'99 IDS Database. *Fifth International Conference on Communication Systems and Network Technologies*, 1-5.

Lee, J., & Park, K. (2019). GAN-based imbalanced data intrusion detection system. *Personal and Ubiquitous Computing*, 1.

Martínez Torres, J., Iglesias Comeseña, C., & García-Nieto, P. (2019). Review: machine learning techniques applied to cybersecurity. *International Journal of Machine Learning and Cybernetics volume*, 4-6.

Mayang Sari, I., Rahman Wijaya, D., Hidayat, W., & Kannan, R. (2021). An Approach to Classify Rice Quality using Electronic Nose Dataset-based Naïve Bayes Classifier. *Fifth International Conference on Communication Systems and Network Technologies*, 1-5.

Mehmood, T., & Md Rais, H. (2016). Machine Learning Algorithms In Context Of Intrusion Detection. *3rd International Conference On Computer And Information Sciences (ICCOINS)*, 1-3.

Ontivero-Ortega, M., Lage-Castellanos, A., Valente, G., Goebel, R., & Valdes-Sosa, M. (2017). Fast Gaussian Naïve Bayes for searchlight classification analysis. *NeuroImage*, 1-3.

Özgür, A., & Erdem, H. (2016). A review of KDD99 dataset usage in intrusion detection and machine learning between 2010 and 2015. *PeerJ Preprints*, 1-3.

Quinlan, J. R. (1996). Learning Decision Tree Classifiers. *ACM Comput. Surv*, 71-72.

Sharafaldin, I., Habibi Lashkari, A., & Ghorbani, A. (2018). Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. *Canadian Institute for Cybersecurity (CIC), University of New Brunswick (UNB), Canada*, 1.

Singh, S. (2020, 01 12). *Building an Intrusion Detection System using KDD Cup'99 Dataset*. Retrieved from Medium: https://medium.com/analytics-vidhya/building-an-intrusion-detection-model-using-kdd-cup99-dataset-fb4cba4189ed

Sreenivasa, S. (2020, 10 12). *Radial Basis Function (RBF) Kernel: The Go-To Kernel*. Retrieved from Towards Data Science: https://towardsdatascience.com/radial-basis-function-rbf-kernel-the-go-to-kernel-acf0d22c798a

Srivastava, D., & Bhambhu, L. (2009). DATA CLASSIFICATION USING SUPPORT VECTOR MACHINE. *Journal of Theoretical and Applied Information Technology*, 1-7.

Thakkar, A. (2020). A Review on Machine Learning and Deep Learning Perspectives of IDS for IoT: Recent Updates, Security Issues, and Challenge. *Archives of Computational Methods in Engineering*, 1-21.

Vaca, C., Riofrío, D., Pérez, N., & Benítez, D. (2020). Buy & Sell Trends Analysis Using Decision Trees. *IEEE Colombian Conference on Applications of Computational Intelligence (IEEE ColCACI 2020)*, 1-7.

Zulhilmi, A. (2021). A Comparison of Three Machine Learning Algorithms in the Classification of Network Intrusion. *Advances in Cyber Security*, 313-324.