DALHOUSIE UNIVERSITY

CSCI4155 MACHINE LEARNING

Machine Learning for Network Intrusion Detection Systems

Project Report

by Group 13

Full Name	Banner ID	Email
Corentin Goetghebeur	B00945496	cr453043@dal.ca
Gabriel Marchand	B00929349	gb614643@dal.ca
Rinchen Toh	B00944448	rn835427@dal.ca

Table of Contents

I. Abstract

Preventing cyberattacks has become a major issue for industrial and governmental bodies in the last 20 years. Able to detect and report fraudulent requests, Intrusion Detection Systems (IDS) are implemented on networks to protect them from potential hacks.

In this project, we aim at using machine learning algorithms in order to construct a model capable of identifying correctly any malicious activities. Usually used in such projects, we build our models to work on the KDD CUP'99 dataset.

Our focus is, through the comparison of multiple machine learning models, to get the best accuracy possible with a focus on good recall scores to avoid, at all cost, possible fraudulent cyberattacks that pass through our detection system.

II. Introduction

Cybersecurity has always played a crucial role in safeguarding one's data against theft and loss. It encompasses everything that relates to protecting our data from cyber attackers who illegally retrieve our information at the expense of causing harm.

The field of application is immense. Any system that requires any sort of data safety and stability or protection of sensitive information will have to implement an intrusion detection system. Therefore, companies, government agencies and even individuals will implement such detection protocols in their operational systems. For instance, antivirus software are accessible and sold to the general public to protect their personal computers.

In this project, we will be focusing on the network protection aspect of cybersecurity.

Network protection is mainly built upon two processes:

1. An Intrusion Detection System (IDS) is a monitoring system that generates an alert when abnormal and suspicious network activities are detected. This system works using either:
    a. Signature Matching which matches known attacks using large libraries of data.
    b. Anomaly-Based Detection which analyzes the usual functioning of the network to then spot and report unusual activity.
2. An Intrusion Prevention System (IPS) acts upon the alert provided by the IDS, to either drop or treat the incoming attack.

For this project, we will only consider the IDS system, and in particular the anomaly-based detection process to overcome the signature matching limits (maintaining an updated database and unknown attacks). The IPS system will not be employed in this project. Two steps are required:

1. Learning phase during which the system will learn how the system behaves in normal circumstances.
2. Detection phase that will allow the system to look for unusual flow.

III. Literature Survey

We first use Singh [1], Devi [9] and Abdallah [3] reviews of the different algorithms on this dataset to target the metrics and algorithms to choose for our study.

Using the KKD CUP'99 dataset [4] of which the details are explained in the article of Divekar [5] in the *IEEE 3rd International conference on Computing, Communication and Security*, Almseiding and al give, at the *15th international Symposium on Intelligent Systems and Informatics*, numerical results for the following algorithm [6]: Random Forest, K-Nearest Neighbors, Naive Bayes and Logistic Regression.

This is to be compared to the work of different machine algorithms by Sabhnanin [7] and Choudahry [8] for Deep Neural,the latest author gets way better recall results using deep neural networks on other intrusion attacks datasets such as NSL-KDD and UNSW NB15

IV. Problem Statement

Due to the massive increase of global cyber-attacks in recent years, there is a rising need for devices/software that are able to detect and hence prevent any possible intrusions in our computer network systems. In comparison to the first half of 2020, global ransomware — restricts access to a computer system until a ransom is paid — attacks have increased by 151% in similar periods of 2021 according to the Canadian Centre for Cyber Security [9]. This could result in individuals and/or companies being heavily impacted should they decide not to pay the ransom. Some examples include financial burdens which could go up to millions of dollars, loss of data (i.e. data breach) which can be detrimental to a company's reputation. Hence, there is a crucial need for tools that are capable of detecting and processing malicious intrusion on the global network system. In this project, we employ the use of Machine Learning models/algorithms as a solution to the aforementioned problem.

V. Proposed Technique

a. Overview of the proposed system

We will employ the use of the following supervised Machine Learning classification algorithms to test their efficiency in classifying the different types of plausible cyber attacks, and also whether a cyber attack is actually present or not (further elaborated in *part b*).

1. K-Nearest Neighbors
2. Decision Tree
3. Random Forest
4. Logistic Regression
5. Naive Bayes
6. Support Vector Machine (SVM)

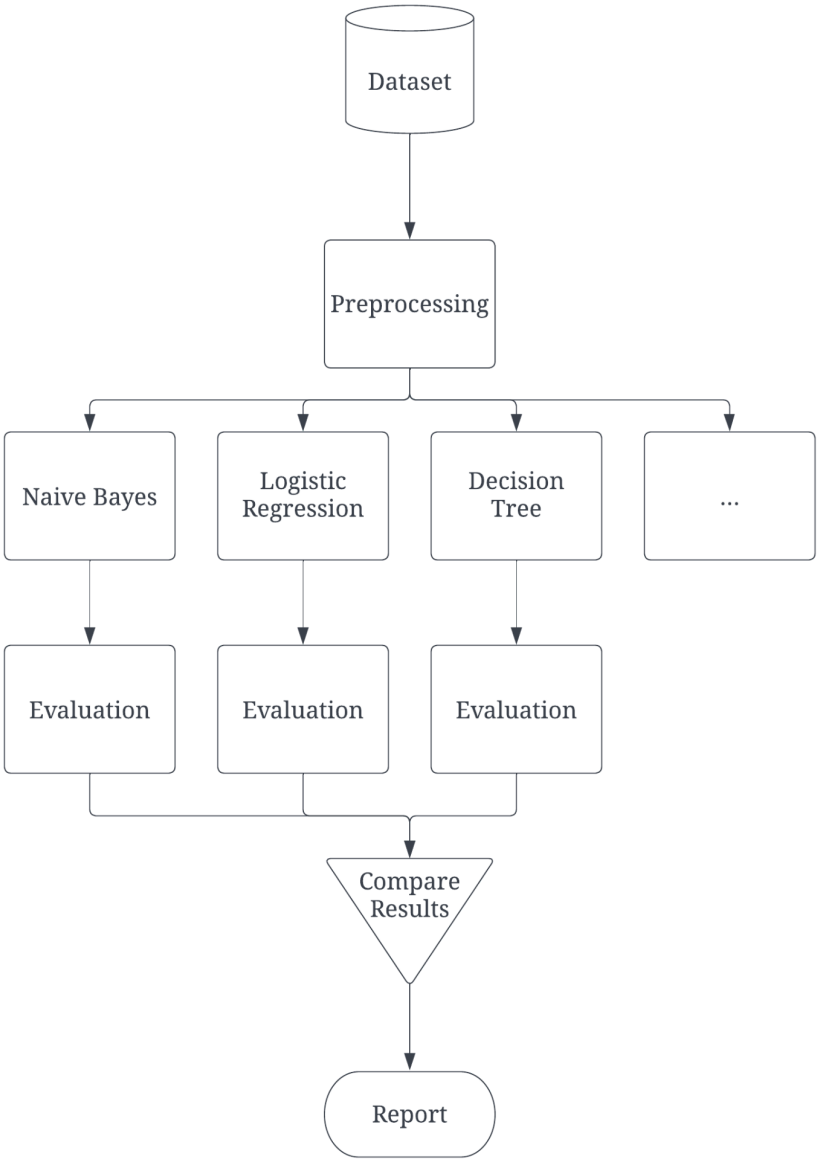

Fig 1: Flowchart of Our Methodology

Before we start running any algorithms, we have to first acquire a dataset and make any necessary changes to it in the pre-processing step to ensure that it is clean and easy to use (refer to Fig 1).

b. Dataset acquisition

We will use the Knowledge Discovery and Data Mining Tools Competition (KDD Cup‘99) dataset, a network traffic dataset widely used for research and academic purposes, this dataset is public domain and can be found on Kaggle indicated in under the ‘References’ section of this report [4].

This dataset has 494,021 rows and 42 columns. The columns include duration, protocol type, number of bytes, whether the user is logged in or not, the number of failed login attempts.

Table 1: Example of a few features and their description

Column/Feature Name	**Description of Feature**
duration	Number of seconds of the connection
protocol_type	Type of protocol used (TCP, UDP, …)
service	Service used in the remote machine (http, telnet, …)
src_bytes	Number of bytes from the source to the destination
…	…
label	Different types of cyberattacks or ‘normal’

The last column of the dataset, ‘label’ contains the type of attack or “normal”.

Here, it is important to note that the ‘label’ column has a total of 22 distinct classes. It is unfeasible and simply ineffective to simply run the classification algorithms. Hence, we have narrowed down to 5 main classes, 4 of which are the main types of cyber attacks that are detected — namely Denial of Service (DOS), Remote to User (R2L), User to Root (U2R), Probing. The last class would represent whether a cyber attack is present or not — indicated as ‘normal’ behavior. This column will be the output of our models.

c. *Preprocessing of data*

Since the dataset is widely used for research purposes, it is already relatively clean, there are no missing values. The first step in the preprocessing was to group together the labels from 22 different classes to 5, the 4 kinds of attacks and the normal behavior. This step is essential because the dataset is very unbalanced, even after the grouping of labels, as seen in *Fig 2*, where the majority of the data has 'Normal' as the label.

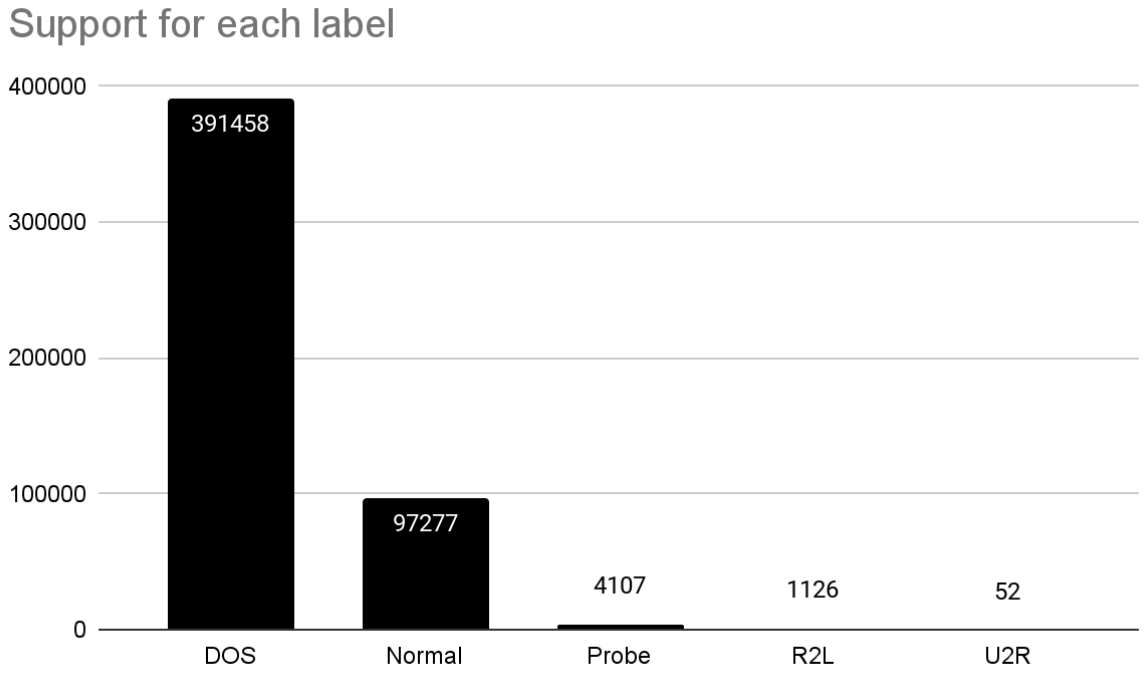

Fig 2: Bar Graph of the Support of Each Label

The dataset has three categorical features (*service*, *protocol_type* and *flag)*, we mapped these features to numerical values. Since the dataset has a lot of columns, we computed the correlation between the features using a correlation heatmap. In the case where features which are highly correlated to one another, we will keep one feature and drop the rest as they do not bring new information to the data and only add to the complexity of our algorithms and possibly result in errors. This allowed us to remove 10 of the columns as part of our dimensionality reduction process.

d. *Splitting Dataset into Training and Testing Data*

We split the data into a training set and a testing set to avoid overfitting problems in our models. We chose to split the dataset into 75% training and 25% testing to have a balance between enough data to train the model and a significant enough testing set. We used the *train_test_split* function from *scikit* library, in particular *sklearn.model_selection*.

VI. Evaluation Metrics

To evaluate our model, we will use the following metrics to choose the best suited model for detecting intrusions in network systems:

a. Confusion Matrix

A summary of prediction results for this classification problem. It gives insights on the types of errors that are made from the Machine Learning model.

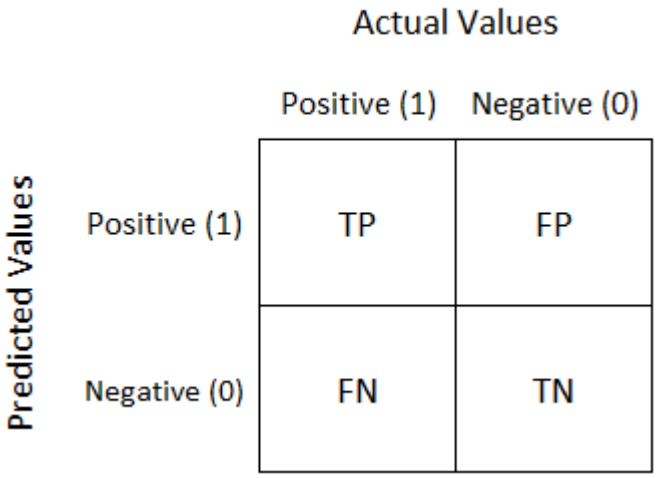

Fig 3: Confusion Matrix Visualization
(taken from My Photoshopped Collection)

With reference to *Fig 3*, the amount of True Positives (TP) represent that the model predictive positive (to be contextualized) and it is true. True Negative (TN) indicates that the model predicted negative (to be contextualized) and it is indeed true. False Positive (FP) indicates that the model predicted positive but it is false. Lastly, False Negative (FN) indicates that the model predicted negative but it is true.

This confusion matrix is vital in measuring the subsequent evaluation metrics that we will be using. However, it is crucial to note that the above confusion matrix applies to only 2 classes. Since there are 5 different classes in this project, the interpretation will be slightly different and calculations for TP, FP, FN and TN are not as straightforward. In this case, we have used the scikit-learn library to aid us.

Nevertheless, a confusion matrix is always a good indicator for the evaluation of a Machine Learning algorithm.

b. Accuracy

Accuracy is the fraction of predictions that the model got right.

However, accuracy as the only evaluation metric alone is not sufficient when working with our class-imbalanced dataset, as there is a significant disparity in support amongst the classes. We will have to draw conclusions together with the other evaluation metrics which are more suited for such class-imbalanced datasets, like precision and recall.

c. Precision

Precision estimates the reliability of the algorithm (i.e. "What proportion of positive identifications was actually correct?")

Since we have more than 2 classes, precision is calculated as the sum of true positives across all classes divided by the sum of true positives and false positives across all classes.

Mathematical formula is as follow (adopted from *machinelearningmastery.com*):

$$precision = \frac{\sum_{C=1}^{5} TruePositives_c}{\sum_{C=1}^{5} (TruePositives_C + FalsePositives_C)}$$

**since we have 5 different classes, summation will start at C = 1 and stop at C = 5.*

It is more appropriate when our focus is to minimize the false positives.

d. Recall

Recall is the ratio of correctly returned data (i.e. "What proportion of actual positives was identified correctly?")

Since we have more than 2 classes, recall is calculated slightly differently. It is the sum of true positives (TPs) divided by the sum of true positives (TPs) and false negatives (FNs) across all classes. Mathematical formula is as follow:

$$recall = \frac{\sum_{C=1}^{5} TruePositives_c}{\sum_{C=1}^{5} (TruePositives_C + FalseNegatives_C)}$$

**since we have 5 different classes, summation will start at C = 1 and stop at C = 5.*

On the other hand, recall is more appropriate if our focus is to minimize false negatives which is strongly applicable in the context of cybersecurity. We would want to minimize occurrence of the model predicting there is no cyber attack on the network systems (i.e. 'normal' behavior) when an attack is actually present. All the dangerous entries should be identified. Hence, the model must produce good recall on the attacks, in particular U2R, R2L and Probe (the DOS attack is only dangerous because of its number).

e. Weighted F1-Score

Weighted F1-Score helps us to have a deeper understanding of the model's accuracy.

It is calculated by taking the average of all F1 scores for each class while considering each class' support. This allows the output average to account for the contribution of each class, especially since we have an unbalanced dataset [10].

VII. Performance Evaluation and Results

To evaluate the performance of the models we developed for this project, we used the *classification_report* and *confusion_matrix* functions from the *sklearn.metrics* library.

a. Classification Report

Since there are a total of 6 different classification reports, one for each model, we have summarized the metrics in the following bar chart below:

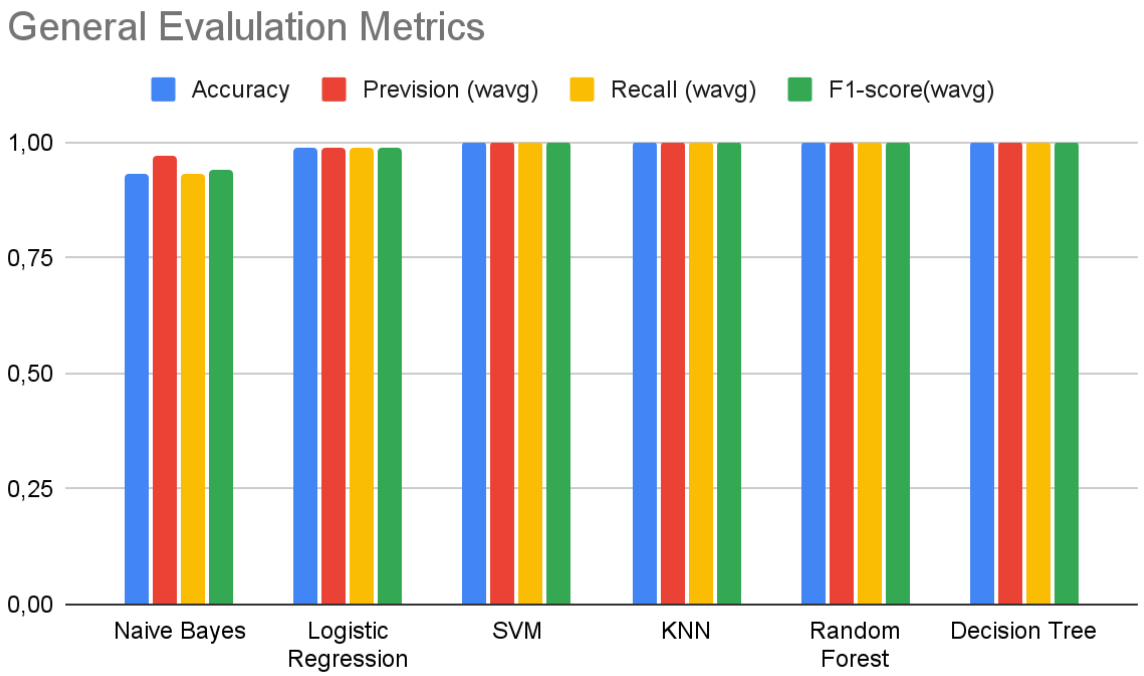

Fig 4: Bar graph of the general evaluation metrics for each algorithm

With reference to *Fig 4*, it is evident that all algorithms achieved good prediction results overall, with high accuracy (most are above 90%), precision, recall and F1-score overall.

However, there are some points to note. As we revisit our primary objective of detecting network threats, the most important labels in the set are *U2R* and *R2L* since they have the most impact if they go undetected. With a class-imbalanced dataset, these specific labels are even more difficult to predict since they have the least support, 0.2% and 0.01% of the dataset respectively. Since all algorithms have excellent overall results, the comparison will be done on the recall on the labels *U2R* and *R2L*.

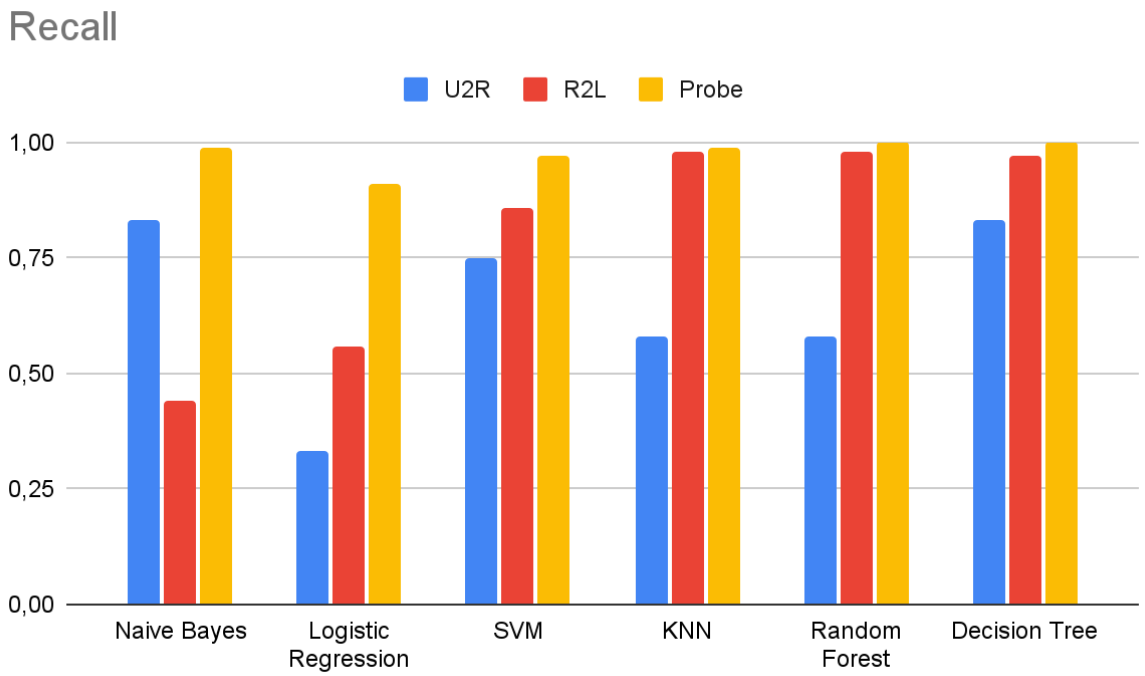

Fig 5: Bar Graph of Recall for the 3 Most Impactful Labels (U2R, R2L and Probe) by the different models

As seen from *Fig 5*, the best performing algorithm based on this metric is Decision Tree, with a recall of 83% for *U2R* and 97% for *R2L*. This is good news to us because the model's ability to detect cyber attacks that are considerably dangerous.

The second best model is SVM with a recall of 75% for *U2R* and 86% for *R2L*.

Naive Bayes also achieved a good recall on the *U2R* label but is really low on the *R2L* label. On the other hand, KNN and Random Forest both achieved a good recall score for *R2L* label but is really low on the *U2R* label.

Logistic Regression is the worst performing algorithm of all, with a lowest recall score of 30% for *U2R* label and barely above 50% for *R2L label.*

Recall for the label *Probe* is generally the same for all models, of more than 90%, which is also good.

Table 2: Classification Report of Decision Tree Model

Class	Precision	Recall	F1-score	Support
DOS	1.00	1.00	1.00	97676
Normal	1.00	1.00	1.00	24482
Probe	0.99	1.00	1.00	1056
R2L	0.98	0.97	0.97	279
U2R	0.67	0.83	0.56	12
Accuracy			1.00	123505
Macro-Avg	0.93	0.96	0.94	123505
Weighted-Avg	1.00	1.00	1.00	123505

The classification report of the Decision Tree model, shown in *Table 2,* proves that it has indeed an incredibly good score on Recall for not only *R2L*, but also on the *U2R* label. Hence, in the context of detecting fraudulent and dangerous cyber attacks, the Decision Tree model performs the best.

b. Confusion Matrix

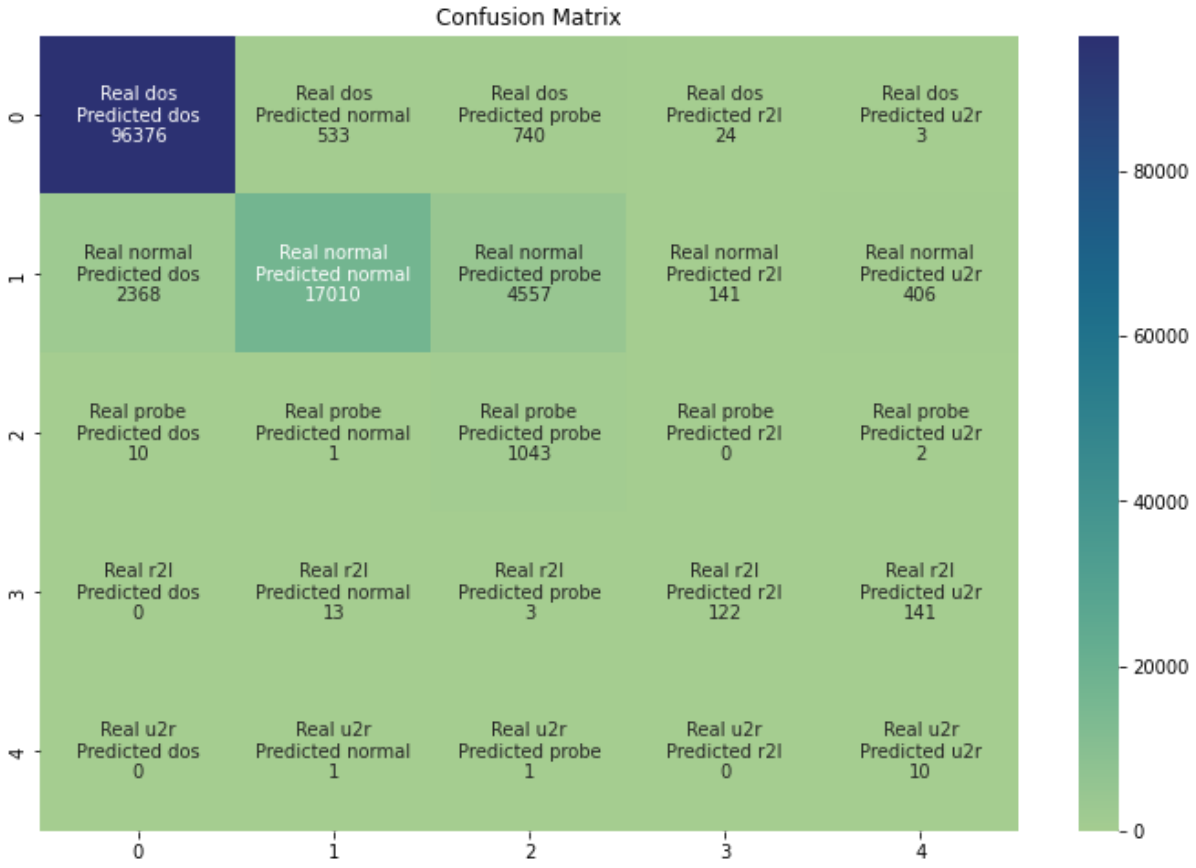

Fig 6: Confusion Matrix for the Naive Bayes Model

The confusion matrix can help to better understand how the model predicted the test values and it also helps to illustrate why some of the metrics are meaningless in our project. For example, we can focus on the *U2R* label for the Naive Bayes algorithm. First, we can see that the recall is satisfactory for this one since out of all the real *U2R* only 2 got predicted as other labels. However, we can also notice that a lot of *Normal* rows were predicted as *U2R*, 406 *normal* predicted as *U2R* is not much compared to the 17010 correct predictions in the *normal* label, but it is enormous compared to the 10 correct predictions of the *U2R* label. The unbalance of the dataset is making the analysis of the results more complicated, and this is why we chose the recall on the *U2R* and *R2L* labels as our main decision metric.

VIII. Conclusion and Future Scope

Our algorithms give very good results for all metrics in detecting the three features Probe, DOS and Normal.

However detecting *U2R* and *R2L* is more complicated due to the very few points available in the dataset, as explained by their low numbers in support. The most important evaluation metric for us is recall, because we aim at correctly detecting the True *U2R* and *R2L* at all costs. If the models fail to do so, this means an attack has gone through the system which could possibly mean danger. To work as a good IDS, we need to avoid any fraudulent request to access our network.

Out of all the machine learning models we've attempted, Decision Tree offers the best option for us as it presents a very high recall for the four first types of attacks and over 75% for *U2R*.

For even better results in detecting fraudulent attacks, some further steps can be taken:

a. ***Clean and more up-to-date versions of dataset***
   KDD-99 is one of the oldest attempts at creating a large dataset of cyber-attacks, it aims mostly at creating NIDS (Network Intrusion Detection Systems) but is suffering from severe flaws that can make it weaker compared to more recent and cleaner datasets [11]. The two main problems are the redundancy of the points given, this makes it a way smaller dataset than it actually is. The second point, which crucially impacts our recall performance, is how much the dataset is skewed with 0.0093 % and 0.005% respectively for the R2L and U2R attack types. In his paper, Divekar [5] gives examples of cleaner and newer datasets that could be used to improve on our results such as the NSL-KDD, which still suffers from drawbacks or the UNSW-NB15 which seems to be the cleanest version of the KDD CUP'99 dataset, for which we removed the most severe defaults.

b. ***Using more sophisticated algorithms***
   This could also improve our metrics significantly: Almseidin [6] outputs the best results using the J48 classifier, which gives a significantly better computation time compared to other algorithms. Although they use more computational power and time, deep neural networks have proven to give better accuracy results on such large datasets [12].

IX. References

[1] P. Singh et A. Tiwari, « A Review Intrusion Detection System using KDD'99 Dataset », International Journal of Engineering Research, vol. 3, nº 11, p. 6.6

[2] R. Devi et M. Abualkibash, « Intrusion Detection System Classification Using Different Machine Learning Algorithms on KDD-99 and NSL-KDD Datasets - A Review Paper », International Journal of Computer Science and Information Technology, vol. 11, p. 65‑80, juin 2019, doi: 10.5121/ijcsit.2019.11306.

[3] E. E. Abdallah, W. Eleisah, et A. F. Otoom, « Intrusion Detection Systems using Supervised Machine Learning Techniques: A survey », Procedia Computer Science, vol. 201, p. 205‑212, janv. 2022, doi: 10.1016/j.procs.2022.03.029.

[4] « KDD99 dataset | Kaggle ». https://www.kaggle.com/datasets/toobajamal/kdd99-dataset

[5] A. Divekar, M. Parekh, V. Savla, R. Mishra, et M. Shirole, « Benchmarking datasets for Anomaly-based Network Intrusion Detection: KDD CUP 99 alternatives », in 2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS), oct. 2018, p. 1‑8. doi: 10.1109/CCCS.2018.8586840.

[6] M. Almseidin, M. Alzubi, S. Kovacs, et M. Alkasassbeh, « Evaluation of machine learning algorithms for intrusion detection system », in 2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY), Subotica, Serbia, sept. 2017, p. 000277‑000282. doi: 10.1109/SISY.2017.8080566.

[7] M. Sabhnani et G. Serpen, « Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context. », janv. 2017, p. 209‑215.

[8] S. Choudhary et N. Kesswani, « Analysis of KDD-Cup'99, NSL-KDD and UNSW-NB15 Datasets using Deep Learning in IoT », Procedia Computer Science, vol. 167, p. 1561‑1573, 2020, doi: 10.1016/j.procs.2020.03.367.

[9] « Canadian Centre for Cyber Security », Canadian Centre for Cyber Security, 24 novembre 2021. https://cyber.gc.ca/en/guidance/cyber-threat-bulletin-ransomware-threat-2021

[10] K. Leung, "Micro, Macro & Weighted Averages of F1 Score, Clearly Explained," Medium, 13-Sep-2022. [Online]. Available: https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained-b603420b292f.

[11] Ravipati, Rama Devi and Abualkibash, Munther, Intrusion Detection System Classification Using Different Machine Learning Algorithms on KDD-99 and NSL-KDD Datasets - A Review Paper (June 2019). International Journal of Computer Science & Information Technology (IJCSIT) Vol 11, No 3, June 2019, Available at SSRN: https://ssrn.com/abstract=3428211 or http://dx.doi.org/10.2139/ssrn.3428211

[12] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, et F. Ahmad, « Network intrusion detection system: A systematic study of machine learning and deep learning approaches », Transactions on Emerging Telecommunications Technologies, vol. 32, nº 1, p. e4150, 2021, doi: 10.1002/ett.4150.

[13] N. Oliveira, I. Praça, E. Maia, et O. Sousa, « Intelligent Cyber Attack Detection and Classification for Network-Based Intrusion Detection Systems », Applied Sciences, vol. 11, nº 4, Art. nº 4, janv. 2021, doi: 10.3390/app11041674.

[14] S. M. Othman, F. M. Ba-Alwi, N. T. Alsohybe, et A. Y. Al-Hashida, « Intrusion detection model using machine learning algorithm on Big Data environment », Journal of Big Data, vol. 5, nº 1, p. 34, sept. 2018, doi: 10.1186/s40537-018-0145-4.

X. Appendix A: Team Contribution

Component	Team Member (who is responsible)
PreProcessing Naive Bayes Logistic Regression Project Proposal Final Report (Proposed Technique, Performance Evaluation, Conclusion and Future Scope) Powerpoint Slides	Corentin
PreProcessing Decision Tree SVM Project Proposal Final Report (Introduction, Evaluation Metrics, Conclusion and Future Scope) Powerpoint Slides	Rinchen
PreProcessing KNN Random Forest Project Proposal Final Report (Abstract, Literature Survey, Conclusion and Future Scope) Powerpoint Slides	Gabriel

XI. Appendix B: Contribution Percentage

Team Member	Contribution (%)
Corentin	33
Gabriel	33
Rinchen	33

----------END OF REPORT----------