VOL. 18, NO. 13, JULY 2023 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

TECHNIQUE FOR PREPARATION OF LARGE DATA FOR MACHINE LEARNING ALGORITHMS TO GENERATE INTRUSION DETECTION SYSTEM

S. P. Senthilkumar and Aranga. Arivarasan,
Department of Computer and Information Science, Annamalai University, Annamalainagar, Tamil Nadu, India
E-Mail: senthil.sp74@gmail.com

ABSTRACT

Machine Learning has become the norm for creating models that predict and detect security breaches in computer systems. Modelling of machine learning systems requires a huge amount of data for training and testing the machine learning algorithm. The present research is concerned with preparing the data prior to feeding it to machine learning algorithms. The necessity for such preparation occurs due to the physical limitations of spreadsheet applications (with regard to the number of records they can handle). This research describes a simple shell scripting approach to combine, extract, and weed out unwanted records and finally prepare the data for feeding to machine learning algorithms.

Keywords: machine learning, security breaches, data preparation, shell scripting, data transformation.

Manuscript Received 30 June 2023; Revised 22 August 2023; Published 13 September 2023

1. INTRODUCTION

The present research is concerned with creating a machine learning model for a dataset provided by the Canadian Institute of Cybersecurity (CIC, https://www.unb.ca/cic/) vide Intrusion Detection System (IDS) Dataset 2017, which is available for download through the page https://www.unb.ca/cic/datasets/index.html (after filling an information sheet about the persons downloading the dataset). The dataset has got 8 different files with a record count of around 2.7 million rows with 79 columns each. On combining the entire dataset into a single file, spreadsheet programs are inadequate to edit the dataset. Hence, it was decided to combine all the records and edit the above data using a shell script on Ubuntu Linux (available in the standard shell environment of any UNIX-compliant system). The task was to rearrange the data in the proper format, assign numeric code to 14 types of attacks enlisted in the CIC dataset, and remove NaN (Not a Number) and Infinity values from the data table. The objective of the present research is to create a single data file containing all the records of the CIC dataset and to make it suitable for supervised machine learning through the substitution of numeric values for human-readable descriptions. The work was completed through several hit-and-fail trial runs.

2. RELATED STUDIES

Ranjit Panigrahi *et al*. conducted a comprehensive analysis of the CICIDS2017 dataset from the CIC (Canadian Institute of Cybersecurity) to evaluate cybersecurity and assess detection and mitigation methods. They identified limitations in existing datasets that could introduce biases into traditional IDS detection systems. Their objective was to propose the development of a merged dataset to overcome these limitations and enhance classification and detection capabilities. The team emphasized the unique characteristics and challenges posed by the dataset during their analysis. [1] Mossa Ghurab *et al*. conducted a thorough study highlighting the importance of utilizing network-based datasets for evaluating intrusion detection methods. Their analysis encompassed several well-known datasets, ultimately recommending the use of contemporary datasets like CIDDS-001, CICIDS2017, and CSE-CIC-IDS2018 for assessing network intrusion detection systems (NIDS). This research emphasizes the significance of carefully selecting suitable datasets to ensure accurate performance evaluation of NIDS. [2] Senthilkumar S.P. *et al*. conducted a study on malicious intrusion attempts using the CIC IDS 2017 dataset. They achieved high accuracy rates of 99.853% and 89.789% with the Random Forest Classifier and Naive Bayes algorithms, respectively. The study emphasized the effectiveness of these supervised learning methods, with Naive Bayes demonstrating fast computational speed and the Random Forest Classifier surpassing previous research. [3] Bhoopesh Singh Bhati *et al*. propose an approach that integrates XGBoost with ensemble-based IDS to improve its performance. By effectively managing the bias-variance trade-off, the approach achieves an exceptional accuracy rate of 99.95% in detecting and preventing intrusions, as demonstrated through experiments with the KDDCup99 dataset. [4] Senthikumar S.P. *et al*. created a refined dataset from the CIC IDS 2017 dataset, consisting of 2.8 million records and 79 parameters. After removing unsuitable records, the dataset's integrity was ensured. Using the Random Forest Classifier (RFC), the analysis resulted in an impressive accuracy rate of 99.853%. [5] Senthilkumar S.P. *et al*. conducted a comprehensive survey on Intrusion Detection Systems (IDS) and evaluated different machine learning (ML) models. After a thorough comparison of their strengths and weaknesses, the RF Classifier method was identified as the most practical choice due to its superior accuracy. [6] In their work, Tianqi Chen *et al*. introduce XGBoost, an advanced tree boosting system renowned for

its exceptional scalability and performance. The authors devise a novel algorithm specifically tailored to efficiently handle sparse data, while also proposing a weighted quantile sketch method for approximate learning. They highlight the importance of cache access patterns, data compression, and sharding in constructing a machine learning system that can scale effectively. By integrating these insights, XGBoost showcases its remarkable capability to tackle real-world problems with minimal resource demands, marking a significant advancement in the field [7]. Yulianto *et al*. conducted a study aimed at improving the performance of AdaBoost-based Intrusion Detection Systems (IDS) using the CIC IDS 2017 Dataset. They employed advanced techniques, including SMOTE, PCA, and EFS. Experimental results showcased the superiority of their proposed method compared to a previous approach. The proposed approach achieved remarkable performance metrics, including high accuracy, precision, recall, and F1 Score. These findings provide substantial evidence of the effectiveness of the novel techniques in advancing intrusion detection capabilities [8]. XGBoost has been identified as a highly recommended classification method by Sukhpreet Singh Dhaliwal *et al*. after an extensive review of multiple studies. Its advantageous features, including adaptability across operating systems, superior accuracy, ability to handle diverse data inputs, provision of both linear model solver and tree algorithms, and successful deployment in industries for processing large volumes of data, have contributed to its widespread acceptance. XGBoost consistently delivers impressive results in various applications. [9] Sivapriya *et al*. Present a novel intrusion detection system in their research paper. The system utilizes the XG Boost algorithm for detecting intrusions. The KDD-99 dataset is employed as input for implementing this approach. The findings of the study reveal that the intrusion detection system, which incorporates the XG Boost algorithm, outperforms other existing algorithms in terms of efficiency and accuracy. [10] Preethi Devan *et al*. Introduced an XGBoost-DNN model that combines feature selection through XGBoost with deep neural networks for network intrusion classification. Their study demonstrated the model's superior performance when compared to existing shallow methods on the NSL-KDD dataset. The model utilizes the Adam optimizer, softmax classifier, and is implemented using Tensor Flow and Python. To validate its performance, cross-validation techniques were employed. Based on the obtained results, a deep learning model consistently achieves a classification accuracy of 97%, surpassing existing models. This model has a competitive advantage over others. The current study focuses on binary classification, but there is potential for future expansion to incorporate multiclass classification. [11] Manish Khule *et al*. assessed various machine learning algorithms and proposed the use of XGBoost, a technique that combines multiple learners, for Intrusion Detection Systems (IDS) using the NSL-KDD99 dataset. Comparative testing showed that XGBoost outperformed SVM in terms of accuracy, leading to its selection as the preferred algorithm. [12] Badisa Naveen *et al*. compared the performance of popular machine learning algorithms in an Intrusion Detection System (IDS) model. Gradient Boosting and XGBoost, belonging to the Boosting Algorithms Family, outperformed other algorithms, indicating their effectiveness for IDS. [13]

3. SHELL SCRIPTING TO HANDLE LARGE CSV DATA

The data contained in the following 8 csv files attributed to the Canadian Institute of Cybersecurity was combined into a single file:

Table-1. CIC IDS 2017 Dataset files.

Sl. No.	File Name
1	Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv (225,745 records)
2	Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv (286,467 records)
3	Friday-WorkingHours-Morning.pcap_ISCX.csv (191,033 records)
4	Monday-WorkingHours.pcap_ISCX.csv (529,918 records)
5	Thursday-WorkingHours-Afternoon-Infilteration.pcap_ISCX.csv (288,602 records)
6	Tuesday-WorkingHours.pcap_ISCX.csv (445,909 records)
7	Tuesday-WorkingHours.pcap_ISCX.csv (445,909 records)
8	Wednesday-workingHours.pcap_ISCX.csv (692,703 records)

The total number of records in the CIC dataset was 2,830,743 (excluding header rows). The first objective was to combine all the dataset files into a single large file containing all the records to easily feed into the machine learning algorithm. The merger of files is an easy process with the help of the cat command. The header rows would be repeated at 8 different locations (representing the header row of each dataset file). The header row would be later removed using the sed command by substituting a blank value against the given row header.

The data contained in the 8 csv files were merged, and numeric replacement values provided through the following script entered in the file datafilter.sh:

```
#!/bin/sh
cat *.csv | sed 's/ //g' | sed 's/BENIGN/1/' | sed 's/Bot/2/' | sed 's/DoSGoldenEye/3/' | sed 's/DoSHulk/4/' | sed 's/DoSSlowhttptest/5/' | sed 's/DoSslowloris/6/' | sed 's/FTP-Patator/7/' | sed 's/Heartbleed/8/' | sed 's/Infiltration/9/' | sed 's/PortScan/10/' | sed 's/SSH-Patator/11/' | sed 's/WebAttack-Brute Force/12/' | sed 's/WebAttack-SqlInjection/13/' | sed 's/WebAttack-
```

XSS/14/' | sed '/DestinationPort/d' | sed '/NaN/d' | sed '/Infinity/d' > data.txt
head -1 M*.csv | sed 's/ //g' > dt.txt && cat data.txt >> dt.txt

The input values were mostly numeric. In some cases, the input values happened to be NaN or Infinity, which corresponded to Not a Number or Infinitely large value. The machine learning algorithms rejected the apparent text input values. The CSV files contained a total of 1,358 NaN values and 2,867 Infinity values. But the datafilter.sh script deleted all rows containing NaN or Infinity value. Supervised machine learning algorithms are limited to comprehending numeric values exclusively. The 79th column of each row contained the classification of the attack type. Columns 1 to 78 were input parameters for machine learning and the 79th column is the prediction parameter. Table 1 depicts the 14 human readable classifications, the numeric replacement provided for the classifications, and the number of rows in which such replacements were found.

Table-2. Attack types, numeric replacements and occurrence count.

Attack type	Numeric replacement	No. of Occurrences
BENIGN	1	2401124
Bot	2	1966
DoS GoldenEye	3	10293
DoS Hulk	4	231073
DoS Slowhttptest	5	5499
DoS slowloris	6	5796
FTP-Patator	7	7938
Heartbleed	8	11
Infiltration	9	36
PortScan	10	158930
SSH-Patator	11	5897
Web Attack - Brute Force	12	1507
Web Attack - Sql Injection	13	21
Web Attack – XSS	14	652

After filtering and preparing the data, the file dt.txt (which held the entire data) contained 2,827,876 rows of data (excluding the header row). So many rows of data cannot be handled by any of the common spreadsheet application programs, which have a physical limit below half of the number of rows contained in the dt.txt file. Finally, the dt.txt file is renamed to dt.csv (to reflect the type of data it contains).

4. XG BOOST

Ensemble learning combines weak learners to improve performance by leveraging their collective strength. Bagging uses random sampling and aggregation of predictions to enhance decision-making. Boosting assigns higher weights to misclassified samples, transforming weak learners into proficient models. Both methods address limitations and improve accuracy. XGBoost, an ensemble additive model, was initially developed by Tianqi Chen during his Ph.D. research at the University of Washington. Renowned for its capacity to handle intricate and high-dimensional datasets, XGBoost stands as a formidable machine learning algorithm. It employs a gradient boosting framework that sequentially builds decision trees to correct errors and improve model performance. XGBoost's optimization strategy uses gradient-based principles to minimize the loss function and determine the optimal model configuration. It is highly efficient in handling large-scale datasets through distributed computing. XGBoost's acclaim comes from its extreme optimization, regularization technique, and its ability to deliver remarkable performance, precision, and scalability in predictive modeling.

$$F = \{ f_1, f_2, f_3, f_4, \ldots\ldots f_m \} \quad \text{set of base learnes}$$

$$\text{Final Prediction } \hat{y}_1 = \sum_{t=1}^{m} f_t(x_i)$$

During the training of machine learning models, the selection of the function employed at each iteration plays a vital role in minimizing the overall loss. Usually, this function, often referred to as the loss function, quantifies the disparity between the predicted and actual outputs. The primary objective is to determine the model parameters that minimize this function, thereby achieving the optimal fit to the training data. The selection of an appropriate optimization algorithm and loss function is crucial, balancing computational efficiency, convergence speed, and the ability to find the best solution. In the mentioned gradient boosting algorithm, the value of ft(xi) for each iteration is obtained by fitting a base learner to the negative gradient of the loss function with respect to the previous iteration's value. In XGBoost, various base learners or functions are considered and the one that minimizes the loss is selected.

$$O = \{x_1, x_2, x_3, x_4, \ldots\ldots x_n\}$$

$$L^{<t>} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{<t-1>} + f_t(x_i)) + \Omega(f_t)$$

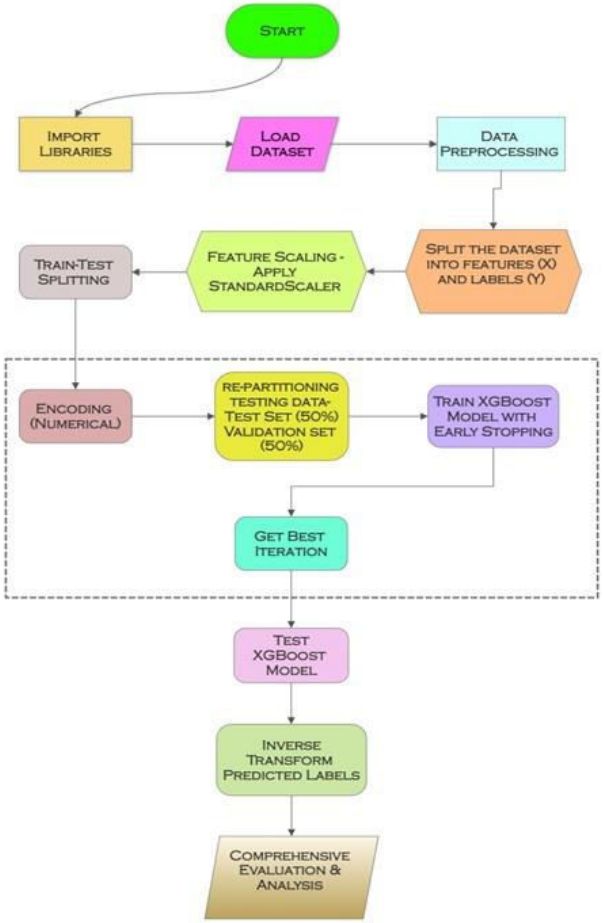

Figure-1. Conceptual framework for the proposed XG boost model.

4.1 Creating Machine Learning Model Using XG Boost Algorithm

Within the realm of machine learning, diverse data types necessitate specific processing techniques tailored to their characteristics. Unstructured data, such as textual or image-based information, finds optimal handling through the utilization of neural networks. In contrast, structured or tabular data, with its well-defined organization, benefits from the superiority exhibited by decision tree-based algorithms. To facilitate the creation of machine learning models, the Python language offers the sklearn library, which provides valuable support in this domain. This encompassing library encompasses functionalities that facilitate the division of data into training and test sets, as well as the assessment of model accuracy.

The provided Python code encompasses a comprehensive approach for the successful execution of a machine learning task. To effectively handle categorical labels, a label encoder was employed, enabling the conversion of such labels into a numerical format, thereby facilitating further analysis. Furthermore, the features underwent standardization using Standard Scaler, ensuring the establishment of a consistent scale across the entirety of the dataset. With a focus on robustness and reliability, the dataset was partitioned into three distinct sets, namely training, validation, and testing. The training set, comprising 80% of the data, served as the foundation for model training, while the remaining 20% was equally distributed between the validation and testing sets. The crux of the code lies within the training of an XGBoost model, skilfully incorporating early stopping techniques to mitigate the inherent risks of overfitting. Continuously monitoring the model's performance on the validation set, the training process would promptly halt if no improvement was detected within a span of 20 consecutive iterations. Subsequently, the model was subjected to evaluation on the previously unseen testing set, utilizing the optimal iteration ascertained through the application of the early stopping mechanism. To holistically gauge the model's performance, a multitude of metrics, including accuracy, precision, recall, and F1-score, were meticulously employed. These metrics, offering invaluable insights into the model's predictive capabilities and its aptitude to generalize to unseen data instances, provided a comprehensive assessment. Additionally, a comprehensive confusion matrix was constructed, enabling a meticulous analysis of the classification outcomes, thus facilitating the detailed examination of both accurate and erroneous classifications for each distinct label. Following the meticulous execution of the code, the evaluation metrics were successfully obtained and subsequently presented through the application of print statements. Furthermore, the Python implementation code produced Table 3, effectively showcasing the model's predictions based on the data available within the "dt.csv" file. With an impressive accuracy of 99.88%, the XGBoost model emerged triumphant. Given that the purpose of this particular XGBoost machine learning model was to test the viability of the dataset extraction file for the creation of numerous other machine learning models, the dataset extraction file will be employed to create a random forest classifier, among other high-accuracy models.

Model Evaluation

[0] validation-mlogloss:0.86206	[21] validation-mlogloss:0.00519	[42] validation-mlogloss:0.00378
[1] validation-mlogloss:0.60043	[22] validation-mlogloss:0.00484	[43] validation-mlogloss:0.00379
[2] validation-mlogloss:0.43065	[23] validation-mlogloss:0.00458	[44] validation-mlogloss:0.00380
[3] validation-mlogloss:0.31345	[24] validation-mlogloss:0.00438	[45] validation-mlogloss:0.00380
[4] validation-mlogloss:0.23015	[25] validation-mlogloss:0.00423	[46] validation-mlogloss:0.00381
[5] validation-mlogloss:0.17005	[26] validation-mlogloss:0.00412	[47] validation-mlogloss:0.00381
[6] validation-mlogloss:0.12624	[27] validation-mlogloss:0.00403	[48] validation-mlogloss:0.00382
[7] validation-mlogloss:0.09418	[28] validation-mlogloss:0.00397	[49] validation-mlogloss:0.00383
[8] validation-mlogloss:0.07063	[29] validation-mlogloss:0.00391	[50] validation-mlogloss:0.00384
[9] validation-mlogloss:0.05326	[30] validation-mlogloss:0.00388	[51] validation-mlogloss:0.00384
[10] validation-mlogloss:0.04041	[31] validation-mlogloss:0.00385	[52] validation-mlogloss:0.00385
[11] validation-mlogloss:0.03092	[32] validation-mlogloss:0.00384	[53] validation-mlogloss:0.00386
[12] validation-mlogloss:0.02390	[33] validation-mlogloss:0.00383	[54] validation-mlogloss:0.00387
[13] validation-mlogloss:0.01871	[34] validation-mlogloss:0.00381	[55] validation-mlogloss:0.00387
[14] validation-mlogloss:0.01488	[35] validation-mlogloss:0.00381	[56] validation-mlogloss:0.00388
[15] validation-mlogloss:0.01203	[36] validation-mlogloss:0.00380	[57] validation-mlogloss:0.00388
[16] validation-mlogloss:0.00992	[37] validation-mlogloss:0.00379	[58] validation-mlogloss:0.00389
[17] validation-mlogloss:0.00835	[38] validation-mlogloss:0.00379	[59] validation-mlogloss:0.00389
[18] validation-mlogloss:0.00718	[39] validation-mlogloss:0.00379	[60] validation-mlogloss:0.00390
[19] validation-mlogloss:0.00632	[40] validation-mlogloss:0.00379	[61] validation-mlogloss:0.00390
[20] validation-mlogloss:0.00567	[41] validation-mlogloss:0.00378	

Table-3. Model evaluation.

Test Accuracy	Precision	Recall	F1-Score
99.880%	**1.00**	**1.00**	**1.00**

4.2 Confusion Matrix Examination

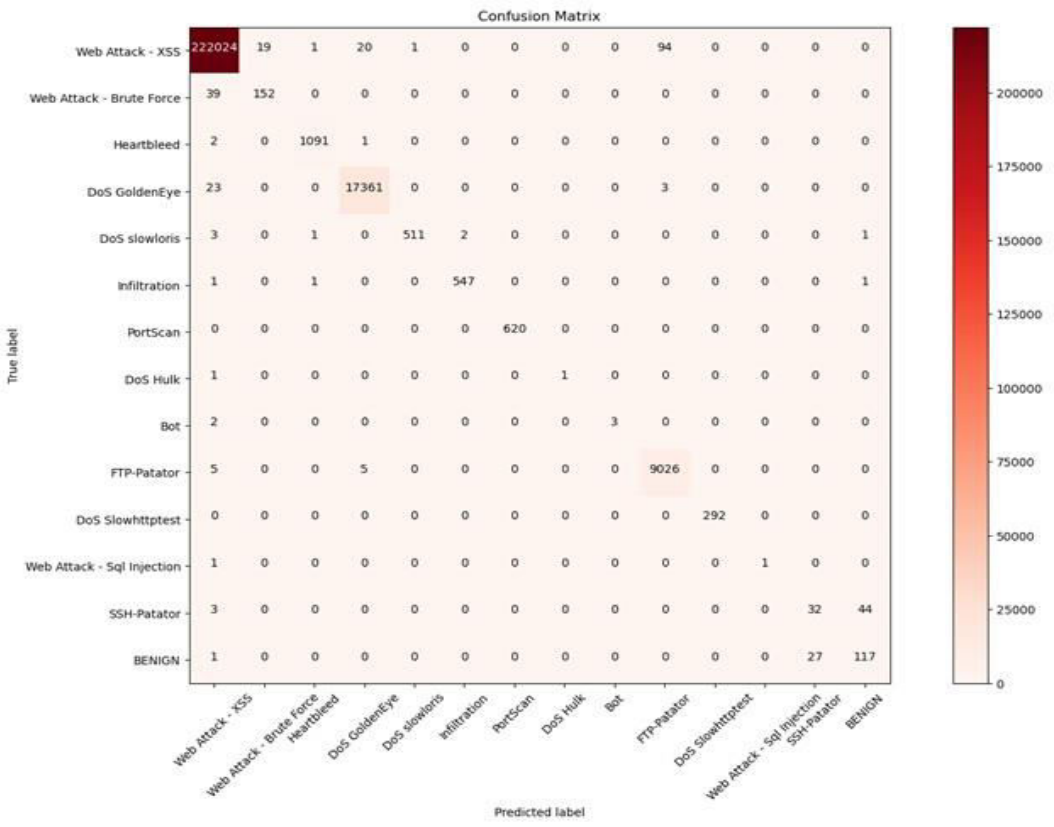

Figure-2. Confusion matrix for XG boost.

Figure-2 illustrates the confusion matrix for the XG Boost classifier. Out of 2, 52,080 samples, 2, 51,778 were correctly classified, while 302 were misclassified. The XG Boost algorithm achieved an accuracy of 99.88019676 %.

4.3 Classification Report Analysis

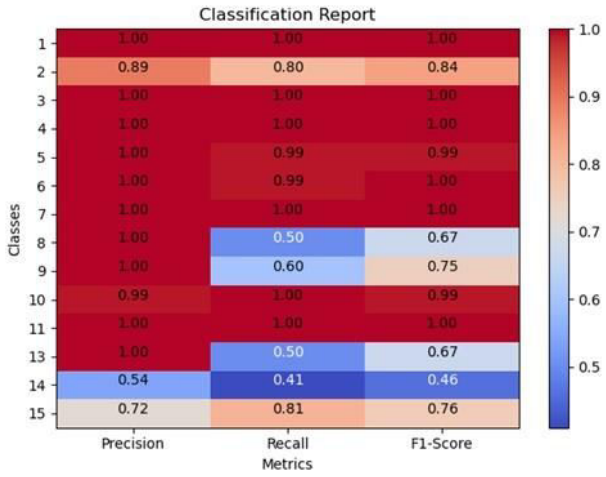

Figure-3. Confusion matrix for XG boost.

5. RESULTS AND DISCUSSIONS

The results produced from the present research are useful for the aggregation of large datasets into single data files and for to use of the data for creating machine learning models. This work has overcome the usual limits with regard to the number of records that can be placed inside a dataset file. Shell scripting is a simple approach and the tools are available in any Unix, Linux, or MacOS operating systems. The shell scripting tools provided a convenient set of commands to aggregate large amounts of data in a single file. The shell command named sed (stream editor) was useful in replacing human readable descriptions with machine readable numeric values. It was also used for removing the 8 header rows, which provided captions for individual CSV files of the CIC dataset. The approach for extraction and filtering of data described in this paper is flexible and can be adapted for any other research involving a huge volume of data (exceeding the physical spreadsheet limit of 1, 048, 576 records). This research identifies sed as the best filter for providing numeric replacements for non-numeric data. In order to ascertain whether spreadsheet applications can handle such a huge amount of data or not, an attempt was made to open the dataset file using a spreadsheet application. Error messages were displayed as shown in Figures 2 and 3 since the data was too large for the spreadsheet application.

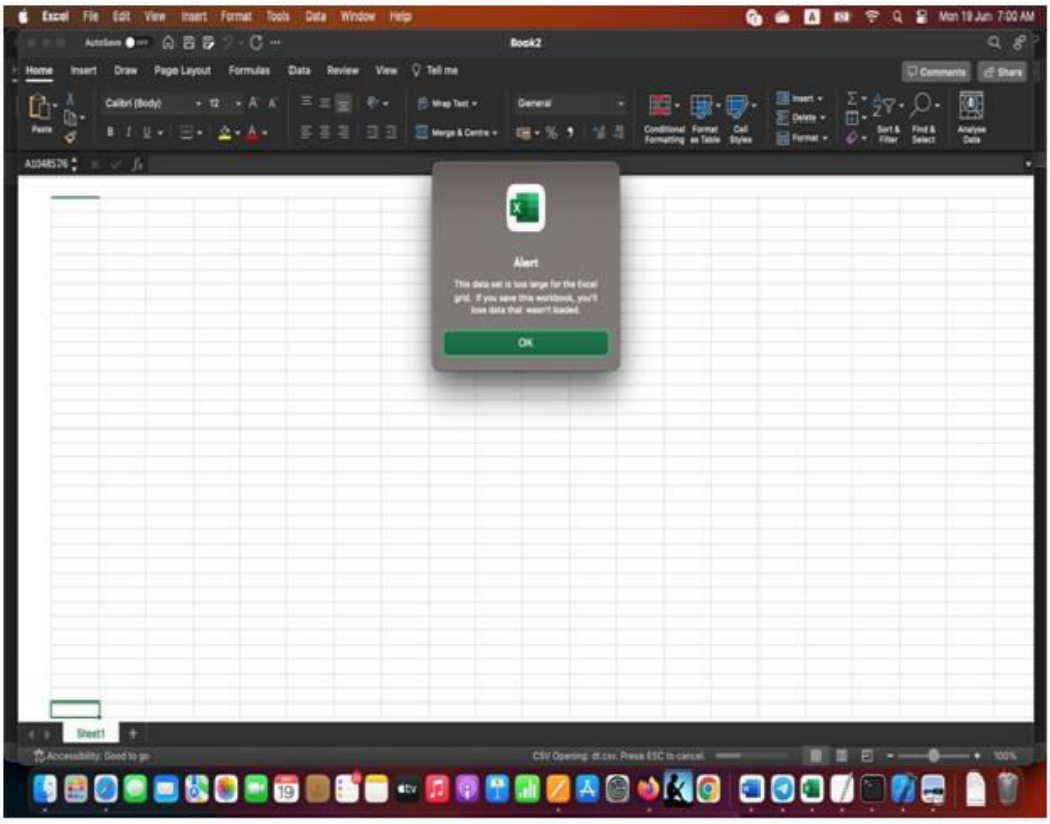

Figure-4. Alert for dataset size.

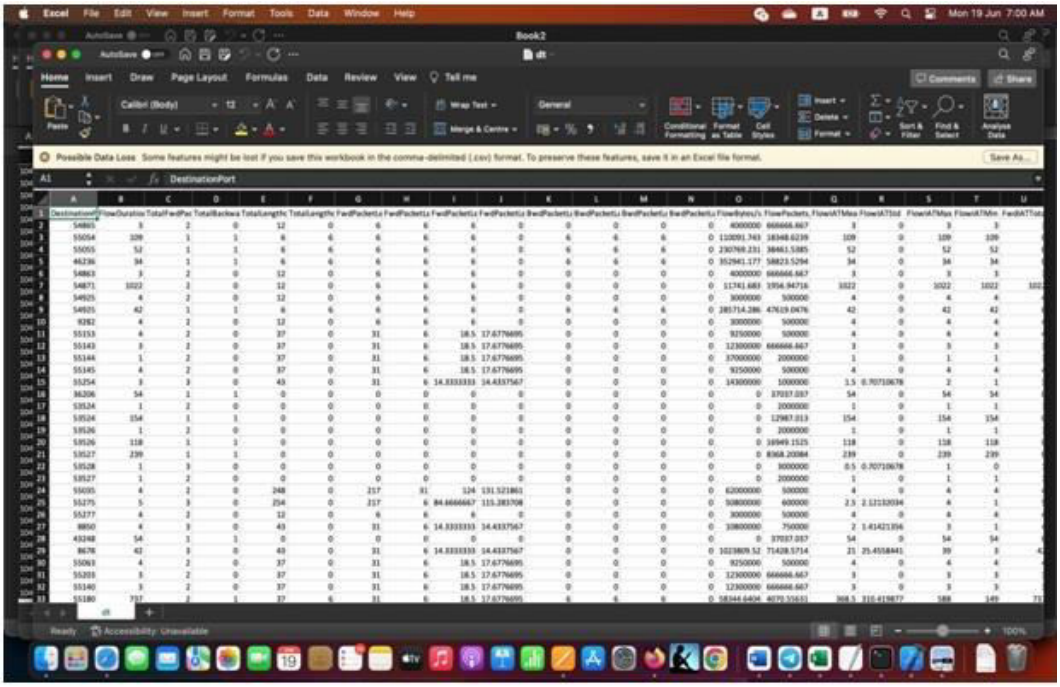

Figure-5. Alert for possible loss of accuracy.

The XGBoost classifier model showcased outstanding classification capabilities, achieving an extraordinary accuracy rate of 99.88%. It also demonstrated impeccable precision, recall, and F1-Score values. By implementing an early stopping mechanism, the model effectively mitigated overfitting and ensured its proficiency in generalizing to unseen instances. When comparing different models, all of them displayed commendable accuracy, perfect precision, recall, and F1-Score values.

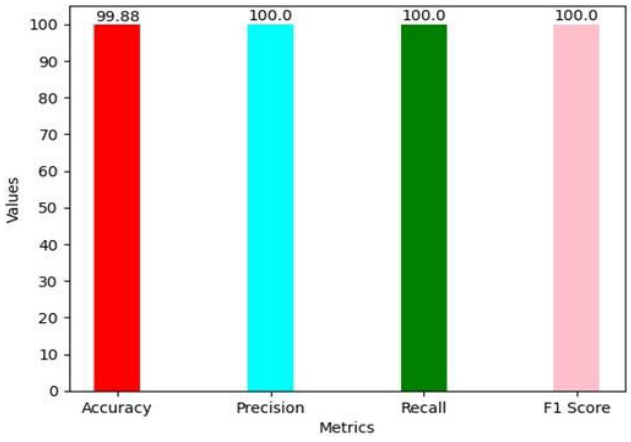

Figure-6. Comparison of Metrics.

6. CONCLUSIONS

The present research was aimed at creating a single aggregate dataset file with a large volume of data derived from the Canadian Institute of Cybersecurity (CIC) dataset. The extraction and filtering were carried out through shell scripting. Using the data extracted from the CIC dataset, an XG Boost model was created and successfully tested. Hence, it is found that the simple data extraction and filtering tools provided in the standard Unix/ Linux shell scripting environment are sufficient for aggregation of data and making it useful for machine learning algorithms.

REFERENCES

Ranjit Panigrahi, Samarjeet Borah. 2018. A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems. International Journal of Engineering & Technology. 7(3.24): 479-482.

Mossa Ghurab, Ghaleb Gaphari, Faisal Alshami, Reem Alshamy and Suad Othman. 2021. A Detailed Analysis of Benchmark Datasets for Network Intrusion Detection System. Asian Journal of Research in Computer Science. 7(4): 14-33, Article no.AJRCOS.66791 ISSN: 2581-8260

Senthilkumar S. P. and Arivarasan A. 2022. An efficient intrusion detection system using machine learning models. Journal of Northeastern University, 25(04), ISSN: 1005-3026. Retrieved from https://dbdxxb.cn/.

Bhati B. S., Chugh G., Al-Turjman F., Bhati N. S. 2020. An improved ensemble based intrusion detection technique using XGBoost. Transactions on Emerging Telecommunications Technologies. e4076. Retrieved from wileyonlinelibrary.com/journal/ett. © 2020 John Wiley & Sons, Ltd. doi: 10.1002/ett.4076.

Senthilkumar S. P., and Arivarasan, A. 2022. Random Forest Classifier implementation of intrusion detection system based on Canadian Institute of Cybersecurity (CIC)

dataset. Advanced Engineering Science. 54(02): 6437-6443.

S. P. Senthilkumar and A. Arivarasan. 2022. Empirical Analysis of Machine Learning Models towards Adaptive Network Intrusion Detection Systems. 2022, 4th International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, pp. 192-200, doi: 10.1109/ICSSIT53264.2022.9716526.

Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. ACM SIGKDD Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA from August 13-17.

Yulianto A., Sukarno P., and Suwastika N. A. 2019. Improving AdaBoost-based Intrusion Detection System (IDS) Performance on CIC IDS 2017 Dataset. In the 2nd International Conference on Data and Information Science (pp. 012018). IOP Conf. Series: Journal of Physics: Conf. Series 1192. IOP Publishing. doi:10.1088/1742-6596/1192/1/012018.

Sukhpreet Singh Dhaliwal, Abdullah-Al Nahid and Robert Abbas. Effective Intrusion Detection System Using XGBoost. Information 2018, MDPI 9(7): 149; https://doi.org/10.3390/info9070149

Siva Priya, Bipin Kumar Sahu, Badal Kumar, and Mayank Yadav. 2019. Network Intrusion Detection System using XG Boost. International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249-8958 (Online). 9(1).

Preethi Devan and Neelu Khare. 2020. An efficient XGBoost–DNN-based classification model for network intrusion detection system. Neural Computing and Applications, 32, 12499-12514. https://doi.org/10.1007/s00521-020-04708-x

Manish Khule and Neha Sharma. 2020. Anomaly detection model based on SVM & XGBoost to detect network intrusions. International Journal of Engineering in Computer Science. 2(2): 07-10.

Badisa Naveen, Jayanth Krishna Grandhi, Kallam Lasya, Eda Mokshita Reddy, Nulaka Srinivasu, and Suneetha Bulla. 2022. Intrusion Detection System (IDS) using Machine Learning Algorithms against Network Attacks. Mathematical Statistician and Engineering Applications, 11081-11090, 71(4).

Why XGBoost? and Why is it so Powerful in Machine Learning: https://abzooba.com/resources/blogs/why-xgboost-and-why-is-it-so-powerful-in-machine-learning/

Code Implementation of XG Boost Model: https://colab.research.google.com/drive/16Db2BJJjAqHyEtoN3k7dJrkXrkf1pNXx?usp=sharing

Enhanced CIC IDS 2017 Dataset: https://drive.google.com/file/d/1WSeQ_rptpOklLydnFWAhs-vou3pEi162/view?usp=sharing