North American Academic Research

TWASP

Research

A few approaches in Encrypted Malware Classifications

RIDOY KUMAR ROY

School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing, China

Abstract: The classification of malware traffic is a critical component of network intrusion detection systems. Because of the recent surge in traffic encryption, it is no longer possible to categorize malware traffic using port-based or signature-based methods. Nowadays, academics and industry developers are turning to learning-based systems for encrypted malware traffic categorization, and mining statistical patterns of traffic behaviors.

Machine learning has been increasingly researched for the detection of malicious network traffic during the last few decades; it is particularly tempting when the traffic is encrypted, as traditional pattern-matching algorithms are ineffective. Several approaches for traffic classification problems have recently been researched with excellent accuracy thanks to the advent of deep learning algorithms. In this research, I will investigate the efficacy of Random Forest, Logistic Regression, and Convolutional Neural Networks for classification tasks.

Keywords: Abnormal traffic; Machine learning; Random Forest; Logistic Regression; Deep neural networks; 1DCNN.

* Corresponding Author: f18030140@njupt.edu.cn

Accepted: 29 November, 2022; **Published**: 18 December, 2022

How to cite this article*: Ridoy Kumar Roy (2022). A few approaches in Encrypted Malware Classifications. North American Academic Research, 5(11), 226-236. doi: https://doi.org/10.5281/zenodo.7456879*

Conflicts of Interest: There are no conflicts to declare.

Publisher's Note: NAAR stays neutral about jurisdictional claims in published maps/image and institutional affiliations.

Introduction

Due to advances in encryption technology, traffic encryption has become widely utilized on the Internet in recent years. Encryption techniques are the principal approach for securing information in a vast variety of businesses and applications. According to Gartner, more than 80% of commercial network traffic will be encrypted by 2019, and 94% of Google network traffic will be encrypted by May 2019. This encryption technique safeguards network users' freedom, privacy, and anonymity while also allowing them to bypass firewall monitoring and surveillance systems [4]. Unscrupulous persons have, however, used encryption to get unlawful gains. In 2020, for example, more than 70% of malware operations employed encryption to conceal malware delivery, instructions, and data leaks. As a result, academics and business have paid close attention to the identification and categorization of encrypted communication [9]. After passing through an encryption algorithm (for example, symmetric cryptography or asymmetric cryptography, etc.), data packets shift from plain-text to cipher-text. A lot of information is lost, which complicates the categorization of encrypted communications. Identifying distinct protocols or application types is frequently required in practical circumstances. This makes traffic categorization more challenging for encrypted application traffic because there are many application kinds and minimal distinction between them.

A variety of techniques have been used in anomaly detection, with data-driven approaches proving to be the most effective in all aspects when it comes to encrypted network traffic, as it has advanced and extended capabilities to better understand the correlation between data or derive deeper meaning from large scale network traffic data. Mixture or multi-level models have been advocated in various studies to improve the efficacy and accuracy of categorizing network data, especially in the detection of anomaly [10-15]. An ensemble approach is a means of learning the relationship between distinct dataset mining strategies, while meta-learning is a method of learning the relationship between these ensemble approaches.

Deep learning methods have recently been demonstrated to be useful in traffic classification, particularly in encryption technology. To do this, DL requires a large amount of labeled data as well as computing capacity. In this article, I'll go through the broad framework for categorizing (encrypted) traffic. For classification tasks such as data collection and cleaning, feature selection, and model selection, I will provide wide guidance. I will also discuss deep learning techniques and their application to traffic classification. Finally, future directions and outstanding issues are addressed. In this particular discovery, I will examine various known ways of classified encrypted malware with real experience.

A. Research Background

Malware has become one of the most serious cyber hazards in recent years, due to the fast expansion of the Internet. Malware is any program that performs harmful operations, such as data leakage, espionage, and so on. Malware, according to Kaspersky Labs (2017), is "a sort of computer software designed to infect a legitimate user's computer and harm it in many ways." Anti-virus scanners are unable to keep up with the rising diversity of malware, resulting in millions of hosts being infected. According to Kaspersky Labs (2016), 6 563 145 distinct hosts were targeted in 2015, with 4 000 000 malware items detected. According to Juniper Research (2016), the worldwide cost of data breaches would reach $2.1 trillion by 2019. Furthermore, the expertise level necessary for malware generation has decreased because of the widespread availability of attacking tools on the Internet nowadays. Due to the widespread availability of anti-detection techniques and the ability to purchase malware on the black market, anybody may become an attacker regardless of skill level. According to recent studies, script-kiddies or automated assaults are becoming increasingly common.

As a result, malware protection of computer systems is one of the most critical cybersecurity jobs for both individuals and enterprises, as even a single assault may result in data breaches and significant losses. The necessity for precise and quick detection systems is necessitated by massive losses and frequent assaults. Current static and dynamic methods do not provide efficient detection, especially when dealing with zero-day attacks. As a result, approaches based on machine learning can be applied. This study explores the optimal feature representation and classification methods, as well as highlights the major aspects and problems of machine learning-based malware detection. According to Zscaler's 2020 Encrypted Attacks Report [1], attacks based on Secure Sockets Layer (SSL) have increased by more than 260 percent, and ransomware has increased by more than 500 percent, when encrypted online traffic was used. As a result, we've seen significant rise in malicious traffic encryption since the COVID-19 period began. Companies are now at higher danger, according to the paper, because existing cyber security solutions are unable to analyze 100% of network traffic. As a result, this is a good moment to look at the detection of encrypted malicious traffic and the usefulness of existing research in dealing with this problem.

Related works

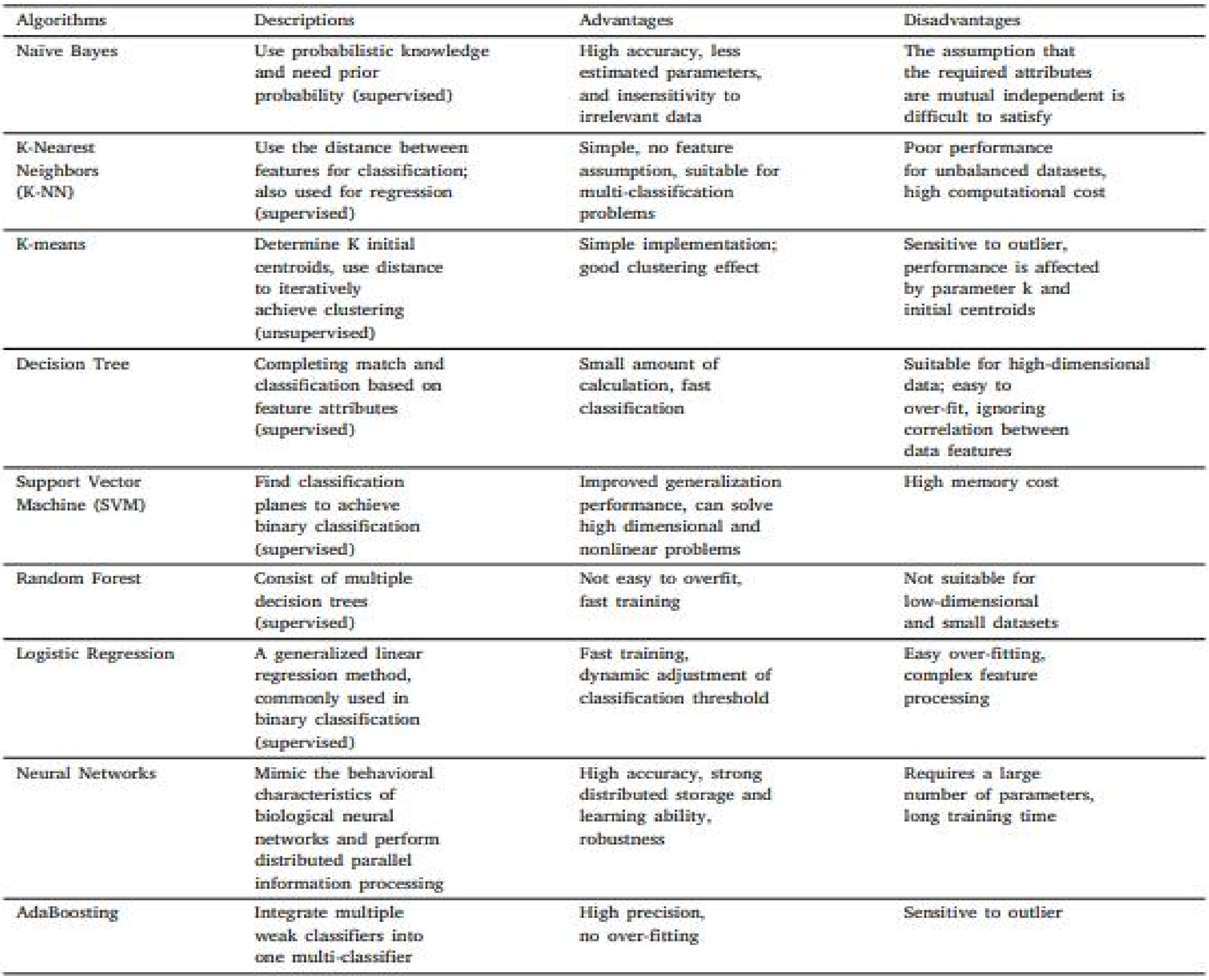
Identifying and recognizing threats in encrypted network communication is extremely tough. However, in recent years, two solutions to this problem have been proposed.

There are currently a few studies on representation learning-based traffic categorization. [2] Gao et al. proposed a Deep belief is used to classify malware traffic. networks. [3] Javaid et al. suggested a malware flow model. Using a sparse auto encoder as an identifying approach. Those Both deep learning techniques and applications were employed in the studies. network-based intrusion detection system design (NIDS). However, they both employed the identical problem in their study. Wang [4] introduced a stacked auto encoder (SAE)-based network protocol identification approach that obtained excellent accuracy utilizing raw traffic data. The duties of traffic categorization and protocol identification are quite similar. As a result, it is fair to believe that the representation learning approach will perform well in the malware traffic classification job. On proxy traffic, Aghaei et al. [5] suggested a classification method based on flow features and a C4.5 decision tree classifier. On both regular encrypted traffic and protocol encapsulated traffic, Draper-Gil et al. [6] suggested a classification approach using just time-related flow parameters. It's worth mentioning that they published a valuable dataset that included both forms of traffic. Using features from host behavior, Huda et al. [7] suggested a semi-supervised technique for detecting unknown attacks on cyber-physical systems (CPS). To update the model and identify unknown attacks, the data without labels and with labels are clustered by Global K-means with cosine similarity at the same time, and the distance between the labeled data and the clustering center is determined. This strategy is effective for combining supervised and unsupervised learning. To overcome the challenge of zero-day malware detection, Kim et al. [8] introduced a static analysis approach called tDCGAN. This approach compares genuine malware data to produce comparable fake malware raw code data in order to detect zero-day malware.

Machine Learning based methods

Scientists can research learning models and algorithms that can aid computers in learning a system from data in the discipline of Machine Learning. To put it another way, one of machine learning's aims is to construct an intelligent system. The two important components that can help machine learning techniques achieve this aim are learning models and learning algorithms. In one form or another, learning models and algorithms are pattern recognition tools.

For correct and efficient traffic categorization, an appropriate traffic classification algorithm is essential. In traffic classification, many machine learning-based algorithms are frequently utilized. Table 1 summarizes and discusses some common machine learning algorithms utilized in existing traffic classification approaches, as well as their benefits and drawbacks.



Algorithms	Descriptions	Advantages	Disadvantages
Naïve Bayes	Use probabilistic knowledge and need prior probability (supervised)	High accuracy, less estimated parameters, and insensitivity to irrelevant data	The assumption that the required attributes are mutual independent is difficult to satisfy
K-Nearest Neighbors (K-NN)	Use the distance between features for classification; also used for regression (supervised)	Simple, no feature assumption, suitable for multi-classification problems	Poor performance for unbalanced datasets, high computational cost
K-means	Determine K initial centroids, use distance to iteratively achieve clustering (unsupervised)	Simple implementation; good clustering effect	Sensitive to outlier, performance is affected by parameter k and initial centroids
Decision Tree	Completing match and classification based on feature attributes (supervised)	Small amount of calculation, fast classification	Suitable for high-dimensional data; easy to over-fit, ignoring correlation between data features
Support Vector Machine (SVM)	Find classification planes to achieve binary classification (supervised)	Improved generalization performance, can solve high dimensional and nonlinear problems	High memory cost
Random Forest	Consist of multiple decision trees (supervised)	Not easy to overfit, fast training	Not suitable for low-dimensional and small datasets
Logistic Regression	A generalized linear regression method, commonly used in binary classification (supervised)	Fast training, dynamic adjustment of classification threshold	Easy over-fitting, complex feature processing
Neural Networks	Mimic the behavioral characteristics of biological neural networks and perform distributed parallel information processing	High accuracy, strong distributed storage and learning ability, robustness	Requires a large number of parameters, long training time
AdaBoosting	Integrate multiple weak classifiers into one multi-classifier	High precision, no over-fitting	Sensitive to outlier

Table 1- depicts the general procedure of classifying network traffic.

The rate at which a classifier is taught is equally important and essential for the algorithm's execution; it indicates how much data the classifier requires to begin performing properly. The ability of a method to be used to a wide range of products is referred to as breadth of application. The ability to update the classifier is particularly important because there is a lot of data and it is difficult to train effectively straight immediately. This is referred to as retraining because

these approaches are frequently utilized by persons who are not experts in the field of data analysis, the algorithm's interpretability is a key consideration.The following qualities were used to assess each of the categories, as shown in Table 2: low, neutral, and high. Each approach has its own set of benefits and drawbacks. A logistic regression is used to get the best findings and indications. As a result, the application was created using this concept.

Algorithm	**Accuracy**	**Scalability**	**Interpretability**	**The complexity of implementation**	**Rapidity**	**Breadth of use**	**Retraining opportunity**
Neural networks	High	Low	Low	High	Low	High	Low

Decision trees	Low	High	High	High	High	High	High
Bayes Method	Neutral	High	Neutral	Neutral	High	Neutral	Neutral
K-nearest neighbor method	Neutral	Neutral	High	Neutral	Low	Neutral	High
Random Forest	High	High	High	Neutral	High	High	High
Logistic Regression	High	High	High	Neutral	High	High	High

Dataset

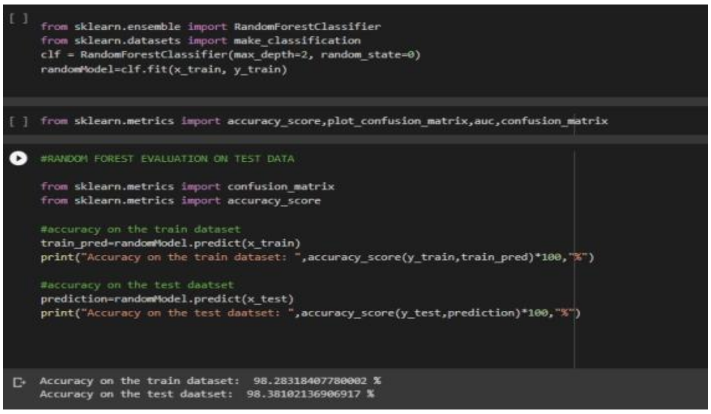
There are a lot of malware datasets available on the internet for analyzing and classifying. These datasets help to develop new machine learning and deep learning models and methods. Here I choose a dataset which includes 41323 binary(exe,dll)-legitimate and 96724 malware files. It also includes features like md5, AddressOfEntryPoint, ImageBase, MajorOperatingSystemVersion, DllCharacteristics, DllCharacteristics, LoadConfigurationSize. This dataset was created for detecting and classifying encrypted malware. I had to do the data processing to process the dataset for training and test.

Features Selection & Classification

Deep Neural Networks are trained using the selected dataset's training set, which comprises a large number of packets separated into windows. The proportions of the training, validation, and testing sets are assumed to be 64 percent, 20 percent, and 16 percent, respectively, for training. I'd also want to point out that I have the best model on the validation set. All features will be delivered to a specially prepared CSV file for additional analysis and processing throughout the random forest model's feature selection. We can also see the 15 top features by opening this CSV file (feat imp.csv). They will be given the number one. These characteristics are also visible in the program console window.

Output & Results of the programs

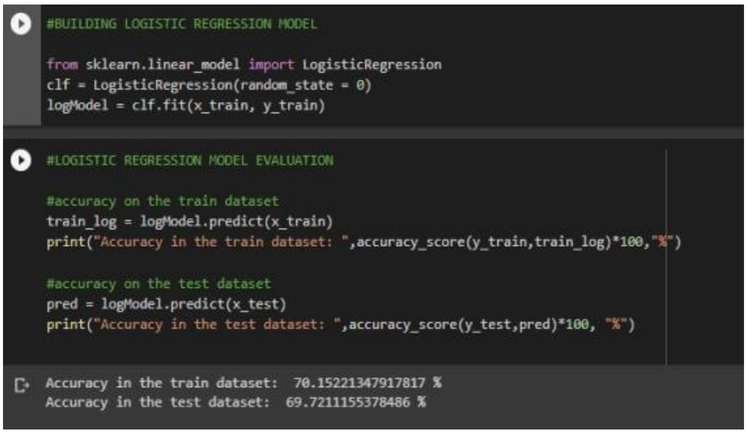
We can see the result after pressing the program's debug button. All features were delivered to a specifically prepared CSV file with 15 top characteristics for further analysis and processing during the feature selection by random forest model. The best features are given a value of one. Also shown in the program console window are the most crucial features. After classifying the data with random forest model, I could find out the accuracy was significantly great. The accuracy was 98.28% for the training dataset and 98.38% for the test dataset. (As shown in figure 5.1)



```
[ ]  from sklearn.ensemble import RandomForestClassifier
     from sklearn.datasets import make_classification
     clf = RandomForestClassifier(max_depth=2, random_state=0)
     randomModel=clf.fit(x_train, y_train)

[ ]  from sklearn.metrics import accuracy_score,plot_confusion_matrix,auc,confusion_matrix

     #RANDOM FOREST EVALUATION ON TEST DATA

     from sklearn.metrics import confusion_matrix
     from sklearn.metrics import accuracy_score

     #accuracy on the train dataset
     train_pred=randomModel.predict(x_train)
     print("Accuracy on the train dataset: ",accuracy_score(y_train,train_pred)*100,"%")

     #accuracy on the test daatset
     prediction=randomModel.predict(x_test)
     print("Accuracy on the test daatset: ",accuracy_score(y_test,prediction)*100,"%")

     Accuracy on the train dataset:  98.28318407780002 %
     Accuracy on the test daatset:  98.38102136906917 %
```

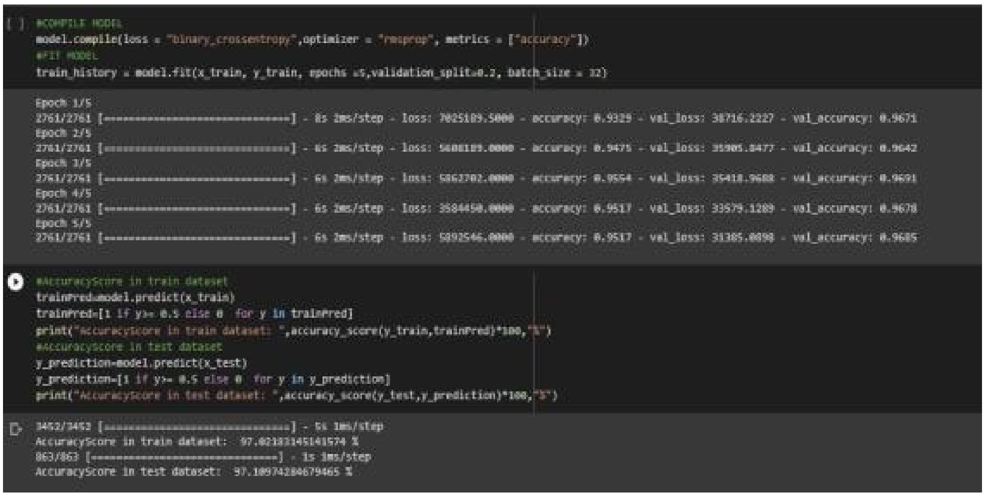
Figure 5.1 Accuracy in Random Forest model

After using the Logistic Regression model to classify the data, I found that the accuracy was not as great as with the other models. The accuracy for the training dataset was 70.15%, while for the test dataset it was 69.72%.



```
#BUILDING LOGISTIC REGRESSION MODEL

from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(random_state = 0)
logModel = clf.fit(x_train, y_train)

#LOGISTIC REGRESSION MODEL EVALUATION

#accuracy on the train dataset
train_log = logModel.predict(x_train)
print("Accuracy in the train dataset: ",accuracy_score(y_train,train_log)*100,"%")

#accuracy on the test dataset
pred = logModel.predict(x_test)
print("Accuracy in the test dataset: ",accuracy_score(y_test,pred)*100, "%")

Accuracy in the train dataset:  70.15221347917817 %
Accuracy in the test dataset:  69.7211155378486 %
```

Figure 5.2 Accuracy in Logistic Regression model

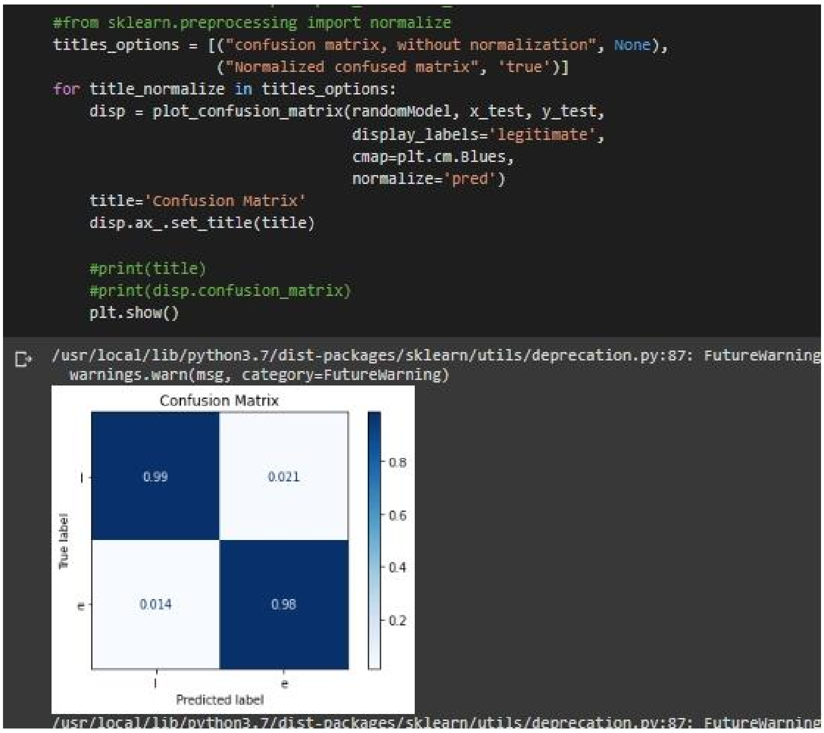


```
[ ]  #COMPILE MODEL
     model.compile(loss = "binary_crossentropy",optimizer = "rmsprop", metrics = ["accuracy"])
     #FIT MODEL
     train_history = model.fit(x_train, y_train, epochs =5,validation_split=0.2, batch_size = 32)

     Epoch 1/5
     2761/2761 [==============================] - 8s 2ms/step - loss: 7025189.5000 - accuracy: 0.9329 - val_loss: 38716.2227 - val_accuracy: 0.9671
     Epoch 2/5
     2761/2761 [==============================] - 6s 2ms/step - loss: 5608189.0000 - accuracy: 0.9475 - val_loss: 35905.8477 - val_accuracy: 0.9642
     Epoch 3/5
     2761/2761 [==============================] - 6s 2ms/step - loss: 5862702.0000 - accuracy: 0.9554 - val_loss: 35418.9688 - val_accuracy: 0.9691
     Epoch 4/5
     2761/2761 [==============================] - 6s 2ms/step - loss: 3584450.0000 - accuracy: 0.9517 - val_loss: 33579.1289 - val_accuracy: 0.9678
     Epoch 5/5
     2761/2761 [==============================] - 6s 2ms/step - loss: 5892546.0000 - accuracy: 0.9517 - val_loss: 31385.0898 - val_accuracy: 0.9685

     #AccuracyScore in train dataset
     trainPred=model.predict(x_train)
     trainPred=[1 if y>= 0.5 else 0  for y in trainPred]
     print("AccuracyScore in train dataset: ",accuracy_score(y_train,trainPred)*100,"%")
     #AccuracyScore in test dataset
     y_prediction=model.predict(x_test)
     y_prediction=[1 if y>= 0.5 else 0  for y in y_prediction]
     print("AccuracyScore in test dataset: ",accuracy_score(y_test,y_prediction)*100,"%")

     3452/3452 [==============================] - 5s 1ms/step
     AccuracyScore in train dataset:  97.02183145141574 %
     863/863 [==============================] - 1s 1ms/step
     AccuracyScore in test dataset:  97.10974284679465 %
```

Figure 5.3 Accuracy in CNN

As we can see in figure 5.3, The accuracy increases in a noticeable amount while using Convolutional Neural Network (CNN). The accuracy was around 97% for both train and test dataset. I have four connected layer dense here and a total of 1057 trainable programs.

Matrixes and graphs verifying the algorithm's accuracy are also included below. It's a table containing four possible combinations of expected and actual values called a confusion matrix (Heatmap). Positive and negative are used to represent predicted values, whereas true and false are used to express actual values. In classification issues, the confusion matrix is used to assess model accuracy.

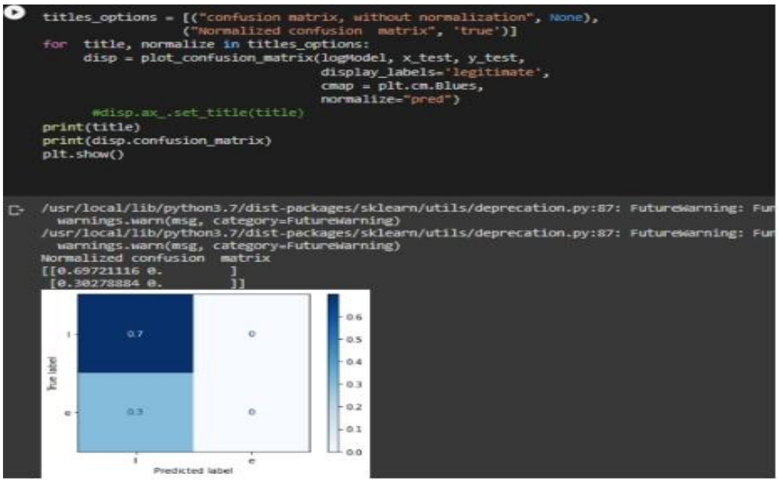


```
#from sklearn.preprocessing import normalize
titles_options = [("confusion matrix, without normalization", None),
                  ("Normalized confused matrix", 'true')]
for title_normalize in titles_options:
    disp = plot_confusion_matrix(randomModel, x_test, y_test,
                                 display_labels='legitimate',
                                 cmap=plt.cm.Blues,
                                 normalize='pred')
    title='Confusion Matrix'
    disp.ax_.set_title(title)

    #print(title)
    #print(disp.confusion_matrix)
    plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning
  warnings.warn(msg, category=FutureWarning)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning
```

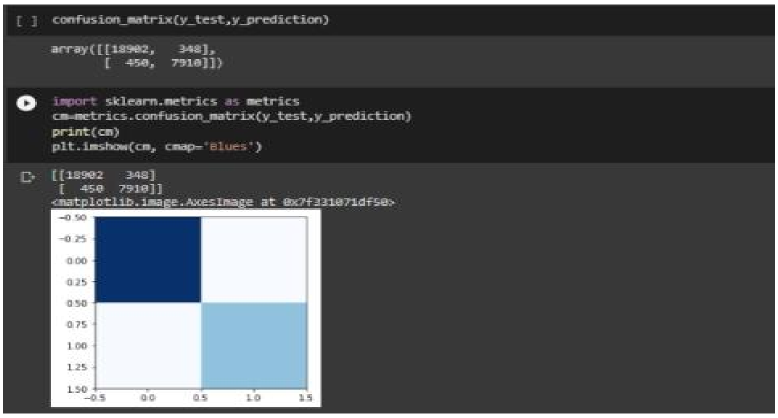
Figure 5.3 Confusion matrix of Random Forest



```
titles_options = [("confusion matrix, without normalization", None),
                  ("Normalized confusion  matrix", 'true')]
for  title, normalize in titles_options:
     disp = plot_confusion_matrix(logModel, x_test, y_test,
                                  display_labels='legitimate',
                                  cmap = plt.cm.Blues,
                                  normalize="pred")
      #disp.ax_.set_title(title)
print(title)
print(disp.confusion_matrix)
plt.show()
```

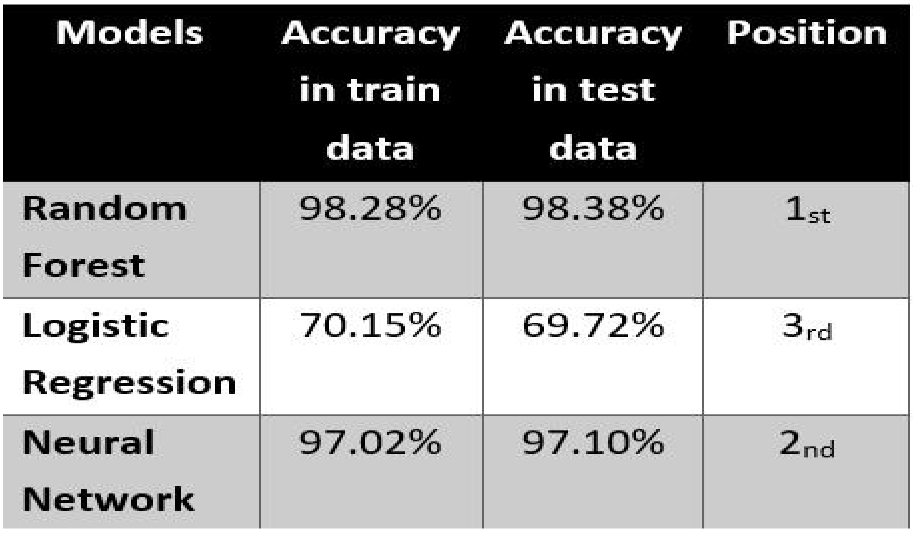
```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Fur
  warnings.warn(msg, category=FutureWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Fur
  warnings.warn(msg, category=FutureWarning)
Normalized confusion  matrix
[[0.69721116 0.        ]
 [0.30278884 0.        ]]
```

Figure 5.4 Confusion matrix of Logistic Regression



```
confusion_matrix(y_test,y_prediction)

array([[18902,   348],
       [  450,  7910]])

import sklearn.metrics as metrics
cm=metrics.confusion_matrix(y_test,y_prediction)
print(cm)
plt.imshow(cm, cmap='Blues')

[[18902   348]
 [  450  7910]]
<matplotlib.image.AxesImage at 0x7f331071df50>
```

Figure 5.5 Confusion matrix of CNN



Models	Accuracy in train data	Accuracy in test data	Position
Random Forest	98.28%	98.38%	1st
Logistic Regression	70.15%	69.72%	3rd
Neural Network	97.02%	97.10%	2nd

Table 3 – Accuracy Comparison of three models

Comparison

When working with the dataset, which comprises 41323 binary(exe,dll)-legitimate and 96724 malware files, I found that combining Random Forest with two one-dimensional Convolutional Neural Networks yielded 98.38 percent accuracy. When dealing with another dataset, which comprises 691,406 packets, the software using the gradient boosting approach and 4 one-dimensional Convolutional Neural Networks achieve around 97% accuracy while Logistic Regression model could score an accuracy of 69.6%. This comparison does not use the same techniques, although they are extremely comparable as all of them were used for classification purpose.

Random Forest, Logistic Regression and Neural Networks are three different learning techniques that can be utilized in similar applications. Random Forest, Logistic Regression are Machine Learning technique, whereas Neural Networks are a Deep Learning technique. In several industry domains, neural networks have been found to outperform a variety of machine learning techniques. They keep learning until the best set of features emerges, resulting in a satisfactory predicted performance. However, a neural network will scale the variables into a series of numbers, making the features indistinguishable to us once the neural network has completed the learning stage.

Neural networks teach a computer how to do a task by evaluating training examples. Because the neural network is loosely based on the human brain, it will have thousands or millions of interconnected nodes. A node can be connected to numerous nodes in the layer below it from which it receives data, as well as several nodes in the layer above it from which it receives data. Each incoming data point is assigned a weight and multiplied and added together. If the weighted total equals zero, a bias is added and then sent to the activation function.

On the other hand, Random Forest is a collection of Decision Trees in which the final leaf node is either the majority class or the average for classification or regression issues. A random forest will produce several Classification trees, and each tree's output is referred to as a 'vote' for that class. The steps for growing a tree are as follows, for each tree, a random sample of rows from the training data will be taken. A subset of features will be picked from the sample taken in the first phase to be utilized for tree splitting. Each tree is expanded to the maximum extent allowed by the parameters until the class votes on it.

The sort of problem we're trying to solve will determine the classifier we should use. The data we're dealing with has an impact on the performance of a classifier model. When the data is structured and we wish to categorize a dependent variable into a certain category, Random Forests are more suitable to be utilized. When the data is large and mostly unstructured, Deep Neural Networks are highly recommended.

Again, Logistic Regression has several benefits as well. They consist of simplicity, adaptability, and the capacity to employ logistic regression in a variety of topic areas. The logistic regression model is intuitive, simple to comprehend, and simple to use from a mathematical perspective. In terms of the analysis, logistic regression is comparable to linear regression. In addition to providing coefficients for the predictor values that determine the score of the dependent variable if the predictor value increases or decreases, hypothesis tests, confidence intervals, and "iterative model-building" to determine the best predictor values to use and those predictor values to remove, logistic regression and linear regression both have similar strengths. Due to the fact that it does not require the rigorous requirements of normality and equality and may fit a greater variety of scenarios than discriminant analysis, logistic regression is a viable substitute. On the other side, the quantity of data needed to construct the model is one of logistic regression's drawbacks. A logistic regression model needs around 50 rows of data per predictor value. When the outcome is dichotomous, logistic regression works well. Although it can be utilized for a multinomial outcome, other modeling strategies could be more appropriate.

As a result, I presume that random forest algorithm is a bit more promising than using Logistic Regression & Convolutional Neural Network in this case. A Random Forest generates accurate forecasts that are simple to comprehend. It is capable of effectively handling huge datasets. In comparison to the decision tree method, the random forest algorithm is more accurate in predicting outcomes. And besides it has a few weak points as well. For instance, when employing a random forest, additional computing resources are required. When compared to a decision tree algorithm, it takes longer period of time to get the job done. The random forest method is a simple and versatile machine learning methodology. It employs ensemble learning, which would allow businesses to address regression and classification issues. This approach is useful for developers since it eliminates the problem of dataset overfitting. It's a highly useful tool for creating accurate predictions in businesses' strategic decision-making.

Conclusion

As a consequence of the work, all tasks were completed and the aim was met: a study of the CNN1D neural network model's suitability for identifying abnormalities in network traffic. The sorts of data abnormalities and methods for detecting them are examined. A categorization of anomaly detection methods is provided. It is examined if recent machine learning algorithms can be used to deal with serial data. Simultaneously, the design, benefits, and drawbacks of deep neural networks, namely the central nervous system, for assessing anomalous traffic were investigated. Folding convolutional layers help us to create hierarchical characteristics and distinguish an assault quickly and correctly. A highest classification accuracy of 98.38 percent was achieved using a random forest classifier. The suggested approach for creating network traffic sequences may be utilized successfully with various neural network models, which could be a subject for future research. At the same time, the development of ideal sequences and the calculation of essential network traffic characteristics are highly dependent on the specific system, which means that a set of heuristic assumptions may be employed for early data processing, considerably improving the results. Implementing a multi-class classification, response mechanism, and preventing the repercussions of assaults using fuzzy logic algorithms is a promising field of research.

Author Contributions: At first page.

Approval: All authors have read and agreed to the published version of the manuscript.
Funding: This research received no external funding.
Institutional Review Board Statement: Not applicable.
Informed Consent Statement: Not applicable.
Data Availability Statement: Not applicable
Acknowledgments: Not Mentioned.
Conflicts of Interest: The authors declare no conflict of interest.

References

[1] Desai, D. 2020: The State of Encrypted Attacks. Zscaler. Retrieved February 24, 2021, from https://www.zscaler.com/blogs/security-research/2020- state-encrypted-attacks

[2] N Gao, L Gao and Q Gao, "An Intrusion Detection Model Based on Deep Belief Networks", Advanced Cloud and Big Data (CBD) 2014 Second International Conference on, pp. 247-252.

[3] A. Javaid, Q. Niyaz, W. Sun and M. Alam. "A Deep Learning Approach for Network Intrusion Detection System." in Proc.9th EAI International Conference on Bio-inspired Information and Communications Technologies. New York, 2016.

[4] Z. Wang, "The Applications of Deep Learning on Traffic Identification." https://goo.gl/WouIM6.

[5] V. Aghaei-Foroushani and A. Zincir-Heywood, "A proxy identifier based on patterns in traffic flows," in HASE, Jan 2015.

[6] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun and A. A. Ghorbani, "Characterization of encrypted and VPN traffic using time-related features", In Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP), pp. 407-414, 2016.

[7] S. Huda, S. Miah, M. M. Hassan, R. Islam, J. Yearwood, M. Alrubaian, and A. Almogren, ''Defending unknown attacks on Cyber-physical systems by semi-supervised approach and available unlabeled data,'' Inf. Sci., vol. 379, pp. 211–228, Feb. 2017. doi: 10.1016/j.ins.2016.09.041.

[8] J.-Y. Kim, S.-J. Bu, and S.-B. Cho, ''Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders,'' Inf. Sci., vols. 460–461, pp. 83–102, Sep. 2018. doi: 10.1016/j.ins.2018. 04.092.

[9] Cisco Encrypted Traffic Analytics 2019. Available online: https://www.cisco.com/c/en/us/solutions/collateral/enterprisenetworks/enterprise-network-security/nb-09-encrytd-traf-anlytcs-wp-cte-en.html (accessed on 10 February 2021).

[10] Most Internet Traffic will be Encrypted by Year End. Here's Why. http://fortune.com/2015/04/30/netflix-internet-traffic-encrypted/, accessed: 2016-03-23.

[11] H. Kim, J. Kim, I. Kim, and T. M. Chung," Behavior-based anomaly detection on big data," in Proceedings of the 13th Australian Information Security Management Conference, 30 November aˆC" 2 December, Edith Cowan University, Western Australia, 2015, pp. 73-80.

[12] R. M. Alguliyev, R. M. Aliguliyev, Y. N. Imamverdiyev, and L. V. Sukhostat," An anomaly detection based on optimization," International Journal of Intelligent Systems and Applications, vol. 12, pp. 87-96, 2017.

[13] H. H. Pajouh, G. Dastghaibyfard, and S. Hashemi," Two-tier network anomaly detection model: A machine learning approach," Journal of Intelligent Information Systems, vol. 48, pp. 61-74, 2017.

[14] R. M. Alguliyev, R. M. Aliguliyev, and F. J. Abdullayeva, "Hybridisation of classifiers for anomaly detection in big data," International Journal of Big Data Intelligence, vol. 6, pp. 11-19, 2019.

[15] S. Aljawarneh, M. Aldwairi, and M. B. Yasin," Anomaly-based intrusion detection system through feature selection analysis and building a hybrid efficient model," Science Journal of Computational, vol. 25, pp. 152-160, 2018.

Ridoy Kumar Roy
School of Computer Science,
Nanjing University of Posts and Telecommunications,
Nanjing, China
f18030140@njupt.edu.cn
WeChat: sagor_0012