Malware Detection Using Neural Networks and Machine Learning Techniques

Velpula Nithin Krishna[1]
Department of Computer Science and Engineering
Amrita School of Engineering, Coimbatore
Amrita Vishwa Vidyapeetham
India
cb.en.u4cse19360@cb.students.amrita.edu

Dr. Senthil Kumar T[*1]
Department of Computer Science and Engineering
Amrita School of Engineering, Coimbatore
Amrita Vishwa Vidyapeetham
India
t_senthilkumar@cb.amrita.edu

Dr. Gireesh Kumar T[*1]
Department of Computer Science and Engineering
Amrita School of Engineering, Coimbatore
Amrita Vishwa Vidyapeetham
India
t_gireeshkumar@cb.amrita.edu

Kartik Srinivasan[*2]
Sr. Architect, IBM Security. IBM India Pvt Ltd
kartiksr@in.ibm.com

Anjali Tibrewal
Sr. Architect, IBM Security.
IBM India Pvt Ltd
anjtibre@in.ibm.com

Sulakshan Vajipayajula[*2]
Architect-CTO Office IBM Security
Bangalore, India
svajipay@in.ibm.com

***Abstract*: In today's contemporary cybernated world, malware has been infested throughout every digital device. It is important for us to remain protected from this malware as it can cause harm and damage our computers, laptops, mobiles, and almost every device that runs on electricity without the approval of the proprietor. It is indeed a universal problem that needs to be tackled. There are many malware detectors that help us in eradicating malware to an extent. In this paper we have utilised the NSL-Knowledge discovery in databases dataset handed down by the University of New Brunswick and implemented various prototypes to predict such attacks, such as basic machine learning techniques which include KNN, Decision Tree, and one of the most powerful algorithms to ever exist, XGBoost, and moved further to implement a deep learning model that is ANN to tackle the challenge and compare the performance metrics. An Apriori model also has been implemented which searches for a series of most occurring sets of items in the dataset. It builds on associations and correlations between the item-sets. A meticulous analysis of various strategies is tendered. We finish the paper by stating our observations and conclusions.**

Keywords: Knowledge discovery in databases , Malware, ANN, KNN, XGBoost, Apriori, Decision Tree.

I. Introduction

With the abundance of malware on the Internet, malware detection is vital because it serves as an early premonition system for computer security against malware and cyber threats. It protects the users from the hackers from gaining access to the computer and prevents information from being compromised. Malware is a malicious application that masquerades as a genuine program in order to infect the computer. Phishing emails, bogus installers, infected attachments, and phishing websites are the most prevalent methods installed on the devices we use.

Hackers disguise malware in order to persuade victims to install it. Because the application appears to be normal, consumers are frequently unaware that it is malware. That is, in essence, how malware is placed on a computer.

Malware lurks in many directories on the computer once it has been installed. It can directly access the operating system if it's a sophisticated sort of malware. Then it begins to encrypt files and keep track of personal data.

II. Roadmap

The following paper is strategized out as follows. In Section 3, we explore the related work that has been done prior, in Section 4, we tender the background about malware prototypes and show the results we've obtained. In section 5, we give a detailed analysis of our findings. In Section 6, we formally present our machine learning and deep learning results with their implementations and desired outcomes.

III. Related Work

This section contains information regarding the prior works done using the same dataset in different research papers over time. The NSL-Knowledge discovery in databases dataset is the rarefied variant of the Knowledge discovery in databases cup '99 dataset. The Knowledge Discovery and Data Mining Cup, directed in 1999, included the Knowledge Discovery in Databases '99 dataset, a subclass of the Defence Advanced Research Projects Agency (DARPA) 1998 paradigm dataset. Following the challenge, scientists have broadly utilised the information to prepare and test AI prototypes for a precise Intrusion Detection System . This segment examines pertinent work in displaying interruption discovery frameworks utilising AI (ML) and deep learning (DL) arrangements on the NSL-Knowledge discovery in databases and Knowledge discovery in databases'99 datasets; : the last-mentioned being a better variation of the Knowledge discovery in databases'99. Mukkamala et al. introduced a neural network (NN) and a support vector machine to perform double-edged categorization. They additionally separated 13 significant elements, prepared the two prototypes utilising these extricated properties, and detailed the outcomes. They reasoned that both support vector machines and neural networks gave faultless outcomes, with the support vector machines performing a tad better. Furthermore, the prototype prepared on the 13 extracted attributes showed just marginally reduced precision. Chan et al. analysed combination approaches-larger part vote, Naive Bayes blend, Dempster-Shafer mix, normal, and neural networks on the Knowledge discovery in databases '99 dataset. They just utilised the preparation set from the Knowledge discovery in databases '99 information and ordered among common traffic and six chose denial-of-service attacks. They inferred that the radial basis function neural network (RBFNN) combination method beat the alternative combination strategies. Heba et al. utilised Principal component analysis to pick features in the informational index and diminish the magnitude of the said features. The chosen attributes were then advanced to a support vector machine to lead 5 class categories: normal, root to local, denial of service, probe, and user-to-root. (U2R). Gao et al. presented a deep belief network (DBN) for an Intrusion Detection System prepared on the Knowledge discovery in databases '99 dataset and reasoned that they achieved better results compared to support vector machines and artificial neural networks (ANNs). All the more as of late, presented a cross-breed deep neural network model to lead double and five-class order. They revealed model execution on different datasets that incorporated the Knowledge discovery in databases '99 and NSL-Knowledge discovery in databases. Liu et al. utilised the dataset NSL-Knowledge discovery in databases on a crossover model composed of K-means, Random Forest, and Deep Learning modules. They accomplished a 85.24% exactness on the dataset. One more mixture model talked about by Wang et al. portrayed an SDAE-ELM. Their point was to handle a couple of restrictions of the Deep Learning prototype, for example, the prolonged stretch of time expected to prepare such a prototype. They corroborated that the model performed better contrasted with other Machine Learning techniques. Shone et al. depicted an Intrusion Detection System , a non-symmetric deep auto-encoder (NDAE) and a Random Forest prototype. They utilised the 10% subset from the Knowledge discovery in databases '99 and utilised the whole NSL-Knowledge discovery in databases dataset for assessments. They detailed outcomes on the previous dataset for five-class grouping and the last dataset for 5-class and 13-class characterisation. They got precise outcomes and presented a productive arrangement that diminishes preparing time by up to 98.81%. A famous strategy for characterisation is the utilisation of a Random Forest model. Ahmad et al. thought about a support vector machine prototype, Random Forest prototype, and extreme learning machines for a powerful Intrusion Detection System utilising information from the dataset NSL-Knowledge discovery in databases. They reasoned that extreme learning machines outflank different techniques. It is fundamental to remember that the creators rejected categorical attributes while cleansing the dataset. Negandhi et al. chose significant elements from the NSL-Knowledge discovery in database dataset utilising Gini significance to prepare a random forest prototype. They acquired outcomes with a 99.88% exactness. Many kinds of examinations have been completed by numerous analysts on the NSL-Knowledge discovery in databases dataset utilising various strategies and apparatuses with a widespread target to foster a viable interruption recognition framework. A comprehensive investigation on the NSL-Knowledge discovery in databases dataset utilising different AI procedures is done and is accessible in the WEKA apparatus.A similar report on the NSL-Knowledge discovery in databases informational index with its precursor Knowledge discovery in databases99 cup informational index is concocted in by utilising the Self Organisation Map Artificial Neural Network. A thorough examination on different informational indexes like Knowledge discovery in databases99, GureKnowledge discovery in databases and NSL-Knowledge discovery in databases are made in utilising different information mining based AI calculations like Support Vector Machine, Decision Tree, KNN, K-Means and Fuzzy C-Mean calculations.

IV. Materials and Methods

In this section we will be discussing the various types of viruses, preprocessing of the dataset and the algorithms which we've used and implemented on our dataset.

Malware recognition is quite possibly the most notable sorts of mechanised observing technology, and it comprises algorithms that identify and make preparations for infections, worms, Trojan horses, spyware, and different kinds of malevolent code. Over the years numerous Unsupervised, Supervised and Semi-Supervised AI methods have been utilised for malware detection. Malware discovery and avoidance programming is generally accessible for servers, gateways, client workstations, and cell phones, for certain instruments considering incorporating checking of malware identification programming introduced on different frameworks or PCs. Malware recognition programs for the most part work behind the scenes and give programmed updates to discovery marks or other reference material that is utilised to recognize pernicious code. There are different kinds of malware found across the globe :

A. **Virus**

Viruses as we can tell are a branch of malicious code or software. It is a malignant software that is connected to a record or document that upholds macros to engineer its code and proliferate from one host to another. When the user downloads the virus it will remain lethargic unless and until the document is opened and used. They are intended to disturb a framework's capacity to work. Therefore, viruses can cause critical functional issues and information misplacement.

B. **Worms**

Worms are also a different type of malignant software that quickly reproduces and proliferates to any gadget inside the organisation. Contrary to viruses, worms needn't require host applications to spread across. A worm taints a gadget by means of a downloaded document or an organisation association before it increases and scatters at an increasingly dramatic rate. Like viruses, they can also seriously disturb the tasks of a gadget and be the reason for information misfortune.

C. **Trojan Virus**

Trojans are veiled as supportive programming codes. Yet, when the client downloads it, the Trojan can access raw information and later on alter, block, or erase the information. This can be very detrimental to the performance of the gadget. Dissimilar to typical worms and viruses, Trojans aren't intended to increase in number.

Figure 1 depicts the techniques we've used in the paper to detect malware:

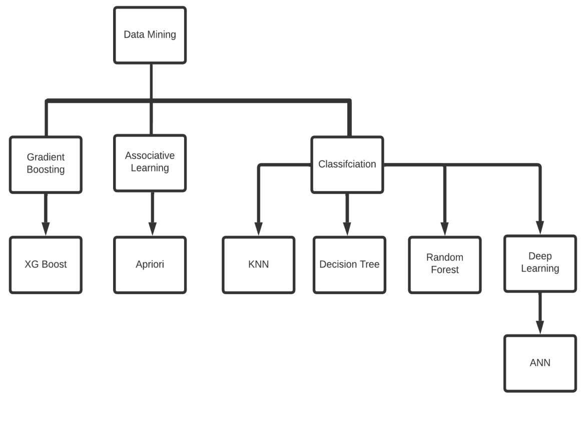


Fig. 1. Taxonomy of Malware detection techniques used in this paper

The proposed model, which includes the accompanying working modules, has been carried out.

A. Dataset Preprocessing

After loading the dataset the first thing we can observe is that the names of the columns are missing so we assign the names to their respective columns and they include – ['duration' ,'protocol_type' ,'service'........,'attack','level']. This dataset has no null values and duplicate values to be found.
We then check for the variance of each feature in the dataset; the features which have 0.0000 variance are to be discarded since they wouldn't be having any effect whatsoever. The feature “num_outbound_cmds” has a variance of 0 and hence has to be dropped from both the Knowledge discovery in databasesTest+.csv & Knowledge discovery in databasesTrain+.csv[1].
Coming to the next part which is correlation , we have plotted a heatmap which tells us which 2 features are highly correlated. The reason for removing highly correlated features is that we will encounter something called the multicollinearity conundrum and our regression model's regression coefficients related to the two highly correlated variables will be untrustworthy.

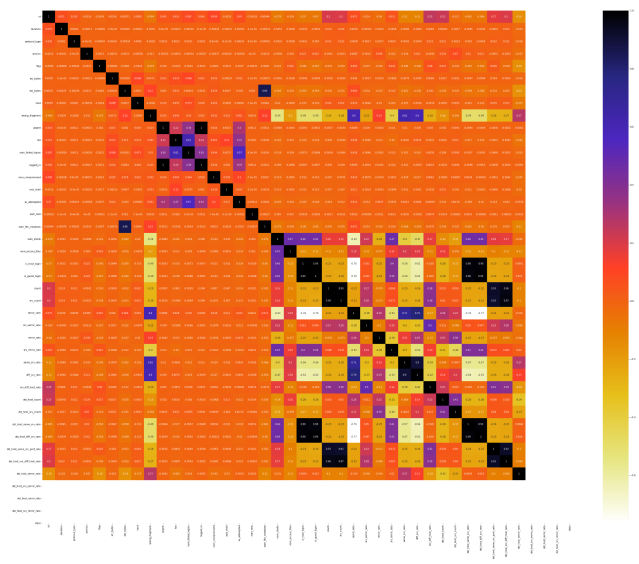


Fig. 2. Heatmap to illustrate the collinearity

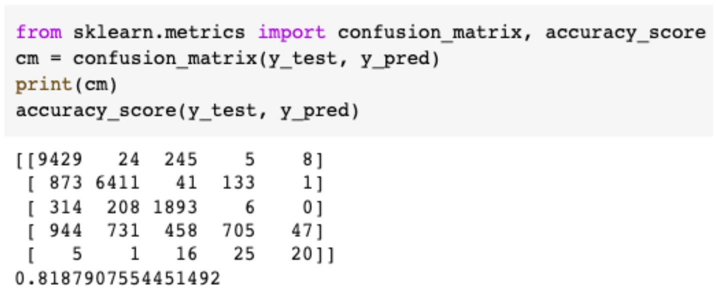
We have set a limit of 0.9 which says that if any 2 features which have either greater than or equal to 0.9 collinearity , those features will be discarded. From our observations we were able to derive which features to be dropped and they are:

- num_root
- srv_serror_rate
- srv_rerror_rate
- dst_host_serror_rate
- dst_host_srv_serror_rate
- dst_host_rerror_rate
- dst_host_srv_rerror_rate

We have then categorised all our attacks into mainly 4 categories and they are denial of service , root to local, probe and u2r.
These attacks have then been converted to numbers , where 0 indicates that it is normal traffic , 1 indicates that it is denial of service , 2 indicates that it is a probe , 3 indicates that it is root to local and 4 indicates that it is u2r.
The categorical features have then been one hot encoded.
Since there is a large variance in the variables with the help of the standard scaler all the values have been scaled to a similar range.

1. K-Nearest Neighbours

In contrast to other supervised statistical pattern recognition malware detection strategies, this is a supervised learning methodology that consistently produces good performance. Distance to find the closest neighbours, some criterion to deduce a categorization from k-nearest neighbour, and the count of neighbours to label the new sample are three important aspects that influence its performance. This approach classifies all transactions that have occurred by computing the least distant point to this particular transaction, and if this least distant neighbour is tagged as malicious, the new transaction is categorised as well. In this circumstance, Euclidean distance is a solid choice for calculating distances. This method is quick and produces fault alerts. Distance metric adjustment can help it perform better. The accuracy of the prediction was found to be 81.8%.



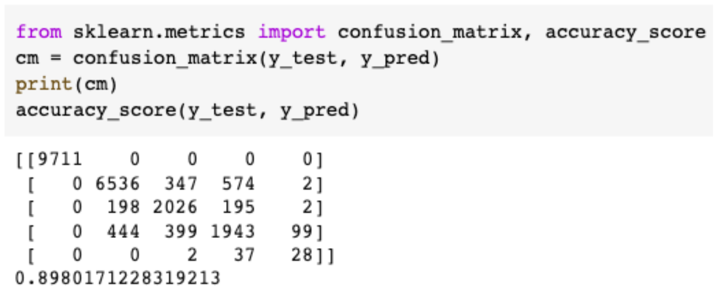
```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[9429   24  245    5    8]
 [ 873 6411   41  133    1]
 [ 314  208 1893    6    0]
 [ 944  731  458  705   47]
 [   5    1   16   25   20]]
0.8187907554451492
```

Fig. 3. Accuracy Scores of KNN Algorithm

2. Decision Trees

For classification and regression, Decision Trees are a non-parametric administered learning strategy. Our point is to take in straightforward choice principles from information credits to create a model that figures the worth of the reliant variable. A tree is a guess to a piecewise steady state. It just requires a little measure of information readiness. Different methods much of the time require information standardisation, the formation of fake factors, and the expulsion of clear qualities. The expense of utilising the tree is logarithmic in the quantity of information used to prepare the tree. Regardless of whether the hidden model from which the information was made abuses a portion of its suspicions, it actually performs well. The accuracy of the prediction was found to be 89.7%.



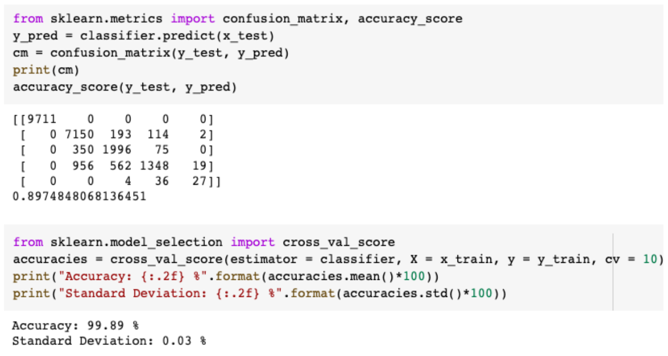
```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[9711    0    0    0    0]
 [   0 6536  347  574    2]
 [   0  198 2026  195    2]
 [   0  444  399 1943   99]
 [   0    0    2   37   28]]
0.8980171228319213
```

Fig. 4. Accuracy Scores of Decision Trees Algorithm

3. XGBoost

XGBoost is quite possibly the most remarkable algorithm to exist in machine learning. It is a decision tree-based outfit Machine Learning calculation that utilises a gradient helping system. Neural organisations outperform any remaining calculations or structures in expectation issues including unstructured information (pictures, text, and so forth). Nonetheless, decision tree-based calculations are currently viewed as top-tier for little-to-medium organised information. It utilises the Gradient Boosting structure to make Machine Learning calculations. It utilises parallel tree boosting to handle a wide scope of information issues rapidly and precisely. The cross validation score of the prediction was found to be 99.89% with a standard deviation of 0.03%.



```
from sklearn.metrics import confusion_matrix, accuracy_score
y_pred = classifier.predict(x_test)
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[9711    0    0    0    0]
 [   0 7150  193  114    2]
 [   0  350 1996   75    0]
 [   0  956  562 1348   19]
 [   0    0    4   36   27]]
0.8974848068136451
```

```
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = x_train, y = y_train, cv = 10)
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

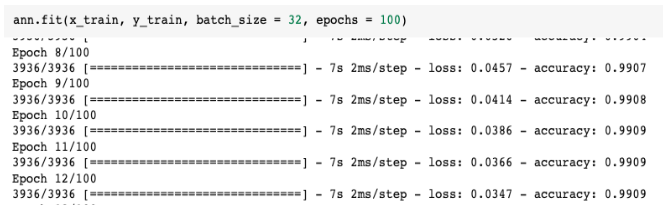
```
Accuracy: 99.89 %
Standard Deviation: 0.03 %
```

Fig. 5. Accuracy Scores of XGBoost Algorithm

4. Neural Networks

A neural organisation is an assortment of neurons that incorporates a perceptron, which is a linear binary classifier that guides in the grouping of approaching information. Input layer, Weights and Bias, Weighted Sum, and Activation Function are the four pieces of a perceptron.

Artificial Neural Network: There are three kinds of layers in an ANN: input, hidden and output. Forward Propagation is the most common way of moving information from the contribution to the transitional hidden layer and in this way to the result layer. The organisation is at first prepared on the malware's standard conduct. Back-proliferated across the organisation, the exchanges that seem, by all devices, to be fake are then ordered as malicious or normal exchanges. This technique has been demonstrated to be exceptionally productive on the grounds that a Neural Network shouldn't be reconstructed, bringing about a high handling speed. The precision of the algorithm is around 99%.



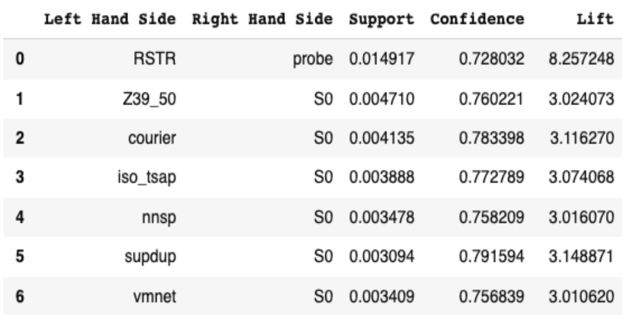
```
ann.fit(x_train, y_train, batch_size = 32, epochs = 100)
```

```
Epoch 8/100
3936/3936 [==============================] - 7s 2ms/step - loss: 0.0457 - accuracy: 0.9907
Epoch 9/100
3936/3936 [==============================] - 7s 2ms/step - loss: 0.0414 - accuracy: 0.9908
Epoch 10/100
3936/3936 [==============================] - 7s 2ms/step - loss: 0.0386 - accuracy: 0.9909
Epoch 11/100
3936/3936 [==============================] - 7s 2ms/step - loss: 0.0366 - accuracy: 0.9909
Epoch 12/100
3936/3936 [==============================] - 7s 2ms/step - loss: 0.0347 - accuracy: 0.9909
```

Fig. 6. Epochs of ANN Algorithm

5. Apriori

Apriori is a mining calculation for affiliation rules. In the datasets, it searches for a progression of regularly occurring sets of components. It depends on the itemsets, connections and relationships. Numerous suggestion administrations utilise this calculation. It works by perceiving the data set's most frequently happening individual things and extending them to greater and bigger thing sets as long as those thing sets happen as often as adequately possible.

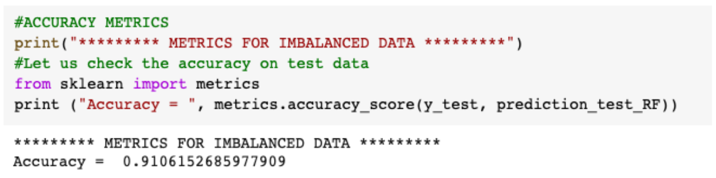


	Left Hand Side	Right Hand Side	Support	Confidence	Lift
0	RSTR	probe	0.014917	0.728032	8.257248
1	Z39_50	S0	0.004710	0.760221	3.024073
2	courier	S0	0.004135	0.783398	3.116270
3	iso_tsap	S0	0.003888	0.772789	3.074068
4	nnsp	S0	0.003478	0.758209	3.016070
5	supdup	S0	0.003094	0.791594	3.148871
6	vmnet	S0	0.003409	0.756839	3.010620

Fig. 7. Table for the most frequent itemsets

6. Random Forest

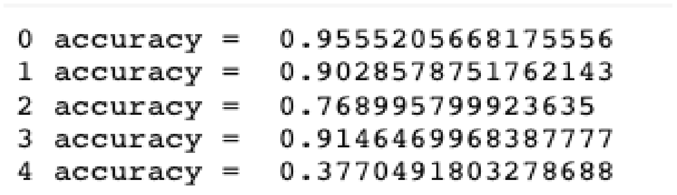
It is built on ensemble learning, for-which it is a procedure of integrating several techniques to solve an intricate task and augment the accuracy of the model. It's a classifier that incorporates countless decision trees on various subcategories of a dataset and estimates the outcomes to enhance the dataset's forecasted score. With every tree being demonstrated unassociated with the other trees, algorithmic productivity of the random forest is relatively superior . The bigger the number of trees in the forest, the more accurate it is and the problem of overfitting is bypassed. The key benefit is that it can handle enormous datasets with high dimensionality and predict output with high accuracy, and it runs efficiently even for large datasets. The accuracy of the prediction was found to be 91.06%. However, the accuracy of the malignant cases was found to be very low and can even be considered to be inconsequential. So, to overcome this problem we have tried upsampling the minorities and increasing their accuracy accordingly.



```
#ACCURACY METRICS
print("********* METRICS FOR IMBALANCED DATA *********")
#Let us check the accuracy on test data
from sklearn import metrics
print ("Accuracy = ", metrics.accuracy_score(y_test, prediction_test_RF))
```

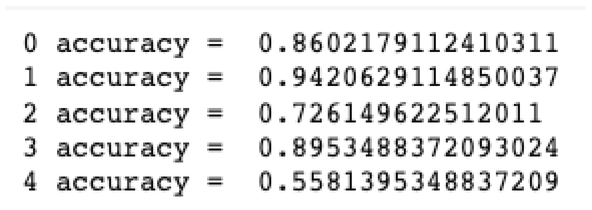
```
********* METRICS FOR IMBALANCED DATA *********
Accuracy =  0.9106152685977909
```

Fig. 8. Accuracy Scores of Random Forest Algorithm



```
0 accuracy =  0.9555205668175556
1 accuracy =  0.9028578751762143
2 accuracy =  0.768995799923635
3 accuracy =  0.9146469968387777
4 accuracy =  0.3770491803278688
```

Fig. 9. Accuracy Scores of all the attacks under Random Forest Algorithm

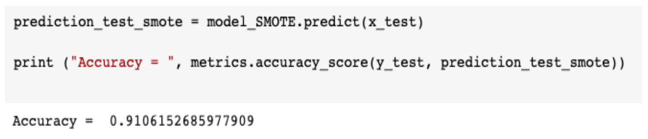


```
0 accuracy =  0.8602179112410311
1 accuracy =  0.9420629114850037
2 accuracy =  0.726149622512011
3 accuracy =  0.8953488372093024
4 accuracy =  0.5581395348837209
```

Fig. 10. Accuracy Scores of the attacks after upsampling the minor cases

7. SMOTE

SMOTE stands for Synthetic Minority Oversampling Technique and it is nothing but an imbalanced classing that incorporates making judicious prototypes on gathering datasets that have an outrageous class lopsidedness. SMOTE first chooses a minority class case indiscriminately and perceives its k closest minority class neighbours. The simulated case is then made by picking one of the k closest neighbours b indiscriminately and associating a and b to shape a line section in the element space. The manufactured examples are produced as an arched blend of the two picked cases a and b.This strategy can be utilised to make as many engineered prototypes for the minority class as are required. As depicted in the paper, it recommends first utilising arbitrary undersampling to manage the quantity of prototypes in the greater part class, then using SMOTE to oversample the minority class to adjust the class conveyance.The collaboration of SMOTE and under-sampling accomplishes superior results when compared to mundane under-sampling.



```
prediction_test_smote = model_SMOTE.predict(x_test)

print ("Accuracy = ", metrics.accuracy_score(y_test, prediction_test_smote))

Accuracy =  0.9106152685977909
```

Fig. 11. Accuracy Scores of SMOTE Algorithm

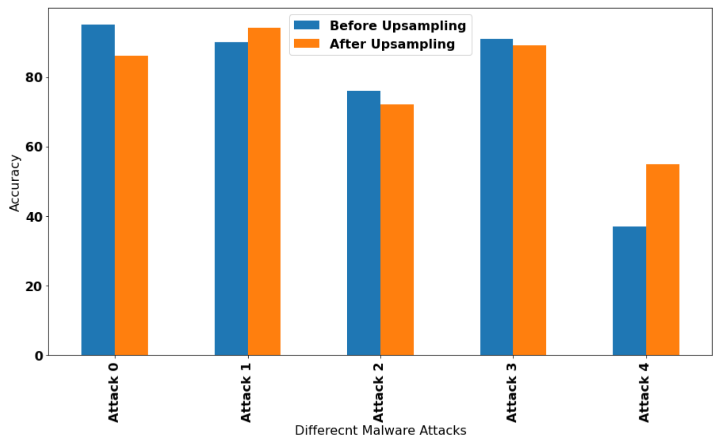


Fig. 12.Accuracy Scores of attacks before and after Upsampling

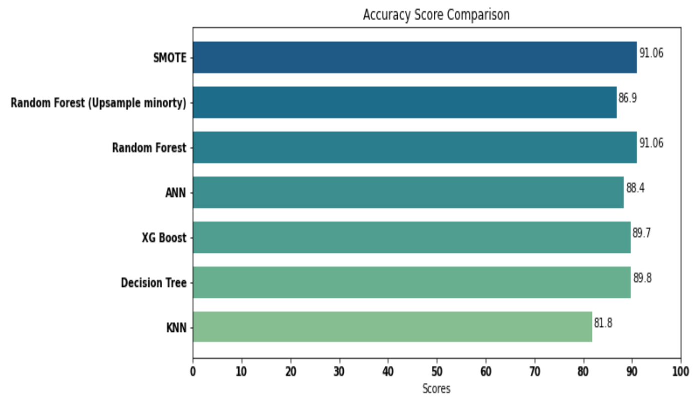


Fig. 13. Accuracy Scores for algorithms implemented

V. Conclusion

Malware is meddlesome programming software that is intended to impair and obliterate PCs and PC frameworks.
Ordinarily, organisations centre around deterrent devices to halt breaks. By securing the peripheral edges, organisations expect they are protected. When some high-level malicious code, in any case, will ultimately advance into your organisation. Subsequently, it is critical to convey advances that constantly screen and recognize malware that has dodged edge safeguards. Adequate progressed malware assurance requires various layers of shields alongside significant organisational perceivability and insight. So we have come up with various algorithms that classify whether the following software is harmful to your device or not.
Dataset Description :
These dataset [1] contain the records of the web traffic seen by a straightforward interruption recognition organisation and are the apparitions of the traffic experienced by a genuine Intrusion Detection System and simply the hints of its reality remains. The dataset contains 43 highlights for each record, with 41 of the elements alluding to the traffic input itself and the last two are names (whether it is a normal or attack) and Score (the seriousness of the traffic input itself).
NSL-Knowledge discovery in databases is an informational index recommended to take care of a portion of the innate issues of the Knowledge discovery in databases '99 informational indexes which are referenced in [6].
Inside the dataset there exist 4 unique classes of assaults:Probe, Denial of Service, Remote to Local, and User to Root(U2R). A short depiction of each assault should be visible underneath [2]: Denial of service is an assault that attempts to close down traffic streams to and from the objective framework. The Intrusion Detection System is overflowed with a strange measure of traffic, which the framework can't deal with, and closes down to secure itself. This keeps ordinary traffic from visiting an organisation.
U2R is an assault that gets going with a typical client record and attempts to get close enough to the framework or organisation, as a super-client (root). The aggressor endeavours to take advantage of the weaknesses in a framework to acquire root advantages/access.
Root to local is an assault that attempts to acquire local admittance to a remote machine. An assailant doesn't have nearby admittance to the framework/organisation, and attempts to "hack" their direction into the organisation.

Probe is a malicious assault that attempts to get data from an organisation. The objective here is to behave like a hoodlum and take significant data, regardless of whether it be close to personal data about customers or banking data.

Classes :	DoS	Probe	U2R	R2L
Sub-Classes :	Apache2 Back Land Neptune Smurf Teardrop Worm Mailbomb Processtable Udpstorm Pod	IPsweep Mscan Nmap Portsweep Saint Satan	Perl Xterm Ps Rootkit Sqlattack Bufferoverflow Loadmodule	Ftp_write Imap Sendmail Httptunnel Spy XSnoop Guess_passwd Snmpguess Phf Named
Total :	11	6	7	10

Table 1. Breakdown of the various subclasses of every attack in the dataset

Albeit these threats exist in the dataset, the conveyance is intensely unproportional. A breakdown of the record dissemination should be visible in the table underneath. Basically, the greater part of the records that exist in every information collection are typical traffic, and the circulation of U2R and root to local traffic are incredibly low. Albeit this is low, this is an exact portrayal of the circulation of cutting edge web traffic assaults, where the most well-known assault is denial of service and U2R and root to local are barely at any point seen.

Dataset	Normal	DoS	Probe	U2R	R2L	Total
KDDTrain +20%	13449	9234	2289	11	209	25192
KDDTrain +	67343	45927	11656	52	995	125973
KDDTest +	9711	7458	2421	200	2654	22544

Table 2. Portrays the attacks scattered across the dataset

The attributes in a traffic record furnish the raw information about the experience with the traffic input by the Intrusion Detection System and can be categorised as the following : Content, Intrinsic, Time-based and Host-based. The following is a depiction of the various classifications of elements:
Content : holds raw details about the primary packets, as they are sent in numerous bits rather than one.
Time-based : holds the investigation of the traffic input for more than a two-second window and contains data like the number of associations it endeavoured to make to a similar host.
Intrinsic : can originate from the header of the bundle without analysing the actual payload, and holds the essential data regarding the packet.
Host-based : analogous to Time-based features, as opposed to breaking down north of a 2-second window, it investigates the progression of associations made.
The features in this dataset can be separated into 4 kinds:

- 6 binary
- 23 discrete
- 4 categorical
- 10 continuous

Flag	Value	Flag	Description
SF	Normal establishment and termination	RSTO	Connection reset by the originator
REJ	Connection attempt rejected	RSTR	Connection reset by the responder
S0	Connection attempt, no reply	OTH	No SYN seen, just midstream traffic
S1	Connection established, not terminated	RSTOS0	Originator sent a SYN followed by a RST
S2	Connection established and close attempt by originator seen	SH	Originator sent a SYN followed by a FIN
S3	Connection established and close attempy by responder seen	SHR	Responder sent a SYN ACK followed by a FIN

Table 3. Description about the flag values

Protocol Type	Service				Flag
Icmp	Other	Urh_i	Time	Private	OTH
Tcp	Link	Ssh	Hostnames	http_2784	S1
Udp	Netbios_s	http_8001	Name	echo	S2
	n	iso_tsap	Ecr_i	http	RSTO
	Smtp	aol	Bgp	ldap	RSTRs
	Netstat	sql_net	telnet	tim_i	RSTOS0
	Ctf	shell	domain	netbios_dgm	SF
	Ntp_u	supdup	ftp_data	uucp	SH
	Harvest	auth	nnsp	eco_i	REJ
	Efs	whois	courier	remote_job	S0
	Klogin	discard	finger	IRC	S3
	Systat	sunrpc	uucp_path	http_443	
	Exec	urp_i	x11	redi_i	
	nntp	rje	imap4	Z39_50	
	pop_3	ftp	mtp	Pop_2	
	printer	daytime	login	Gopher	
	vmnet	domain_cu	tftp_u	Csnet_ns	
	netbios_ns	pm_dump	kshell		

Table 4. Breakdown of the possible values for the categorical features

VI. Summary

In this paper we have constructed a unique methodology for malware detection, where the individuals have been grouped based on their malware activities and derived familiar patterns to initiate an outline for every individual. Then various classifiers have been implemented on the different groups and the performance metrics were produced for every type of classifier. These active alterations in the variables lead the system to change to new activity responses every now and then. We attempted to remove the collinearity which leads to the concept of multicollinearity conundrum that affects the performance of the model. Finally, we discovered that the algorithms XG Boost and ANN produced the best results. We came across a variety of malware detection tools that are now available and many of them were capable of detecting the malicious software , thanks to the development in technology. The drawback of all of the strategies presented thus far is that they only produce accurate results when applied to a certain dataset and, in some cases, only with certain special features.As technology advances, there will be more malevolent software produced and we need to keep updating our algorithms to it. Therefore, we must develop a solution that is as exact and accurate in all conditions and with a variety of datasets. Algorithms such as, ANN, have excellent accuracy and detection rates but are difficult to train and the cost is unscrupulous to train them.We have tried improving the accuracy scores of the neural networks by increasing the number of hidden layers ,many studies have proved that by doing this we may improve our score to an extent but will soon lead us to a saturation point. Also a change in the activation function [3] has been experimented on the model , such as the activation function has been set to Relu, Sigmoid and Tanh and we have found that Relu [4] gives us the best accuracy in our scenario. Another way to increase the performance of ANN is by increasing the number of neurons [5] but just like the hidden layers we can improve it to a certain extent and then we will reach a saturation point. KNN, on the other hand, performs well with small datasets but not with huge datasets & coming to Decision Trees, they perform better on raw, unsampled data.

References

1. M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, “A Detailed Analysis of the Knowledge discovery in databases CUP 99 Data Set,” Submitted to Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), 2009
2. https://www.cisco.com/c/en_in/products/security/advanced-malware-protection/what-is-malware.html
3. M. Shyu, S. Chen, K. Sarinnapakorn and L. Chang, "A novel anomaly detection scheme based on principal component classifier", Proceedings of the IEEE Foundations and New Directions of Data Mining Workshop in conjunction with the Third IEEE International Conference on Data Mining (ICDM03), pp. 172-179, 2003.
4. C. E. Landwehr, A. R. Bull, J. P. McDermott and W. S. Choi, "A taxonomy of computer program security flaws", ACM Comput. Surv., vol. 26, no. 3, pp. 211-254, 1994.
5. J. McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory", ACM Transactions on Information and System Security, vol. 3, no. 4, pp. 262-294, 2000.
6. S. J. Stolfo, W. Fan, W. Lee, A. Prodromidis and P. K. Chan, "Cost-based modeling for fraud and intrusion detection: Results from the jam project", discex, vol. 02, pp. 1130, 2000.
7. R. P. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K. R. Kendall, D. McClung, et al., "Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation", discex, vol. 02, pp. 1012, 2000.
8. S. Axelsson, "The base-rate fallacy and the difficulty of intrusion detection", ACM Transactions on Information and System Security (TISSEC), vol. 3, no. 3, pp. 186-205, 2000.
9. J. Gaffney and J. Ulvila, "Evaluation of intrusion detectors: A decision theory approach", Proceedings of IEEE Symposium on Security and Privacy (S&P), pp. 50-61, 2001.
10. G. Di Crescenzo, A. Ghosh and R. Talpade, "Towards a theory of intrusion detection", Lecture notes in computer science, vol. 3679, pp. 267, 2005.
11. A. Cardenas, J. Baras and K. Seamon, "A framework for the evaluation of intrusion detection systems", Proceedings of IEEE Symposium on Security and Privacy (S&P), pp. 15, 2006.
12. G. Gu, P. Fogla, D. Dagon, W. Lee and B. Skoric, "Measuring intrusion detection capability: An information-theoretic approach", Proceedings of ACM Symposium on Information computer and communications security (ASIACCS06), pp. 90-101, 2006.
13. K. Leung and C. Leckie, "Unsupervised anomaly detection in network intrusion detection using clusters", Proceedings of the Twenty-eighth Australasian conference on Computer Science, vol. 38, pp. 333-342, 2005.
14. D. Ruck, S. Rogers, M. Kabrisky, M. Oxley and B. Suter, "The multilayer perceptron as an approximation to a Bayes optimaldiscriminant function", IEEE Transactions on Neural Networks, vol. 1, no. 4, pp. 296-298, 1990.
15. Vipin Kumar, Himadri Chauhan, Dheeraj Panwar, “K-Means Clustering Approach to Analyze NSL-Knowledge discovery in databases Intrusion Detection Dataset”, International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-3, Issue-4, September 2013
16. Santosh Kumar Sahu Sauravranjan Sarangi Sanjaya Kumar Jena, “ A Detail Analysis on Intrusion Detection Datasets”, 2014 IEEE International Advance Computing Conference (IACC)