Improving Ransomware Detection Techniques using Machine Learning

Usha G.
Department of Software Engineering
SRM Institute of Science and Technology
Chennai, India
ushag@srmist.edu.in

Ishan Chaudhari
Department of Software Engineering
SRM Institute of Science and Technology
Chennai, India
iu7825@srmist.edu.in

Varshini Venkatesh
Department of Software Engineering
SRM Institute of Science and Technology
Chennai, India
vv8781@srmist.edu.in

Abstract*— A challenge that governments, enterprises as well as individuals are constantly facing is the growing threat of ransomware attacks. Ransomware is a type of malware that encrypts the user's files and then demands a huge sum of money from the user. This increasing complexity calls for more advancement and innovative ideas in defensive strategies used to tackle the problems. In this paper, firstly we discuss the existing research in the field of ransomware detection techniques and their shortcomings. Secondly, a juxtaposed study on various machine learning algorithms to detect ransomware attacks is explained. Various behavioural data such as API Calls, Target files, Registry Operations, Signature, Network Accesses were collected for each ransomware and benign sample. These collected samples were used to train Machine Learning Algorithms like KNN, Naïve Bayes, Random Forest, Decision Trees. Further optimization was done using hyperparameters. Finally, we have used the model(s) Accuracy, F1 Score, Precision and Recall to compare the results observed. The acquired result can be used further for the improvement of various antivirus systems in detecting ransomware softwares.

***Keywords*— Ransomware, Detection, KNN, Random Forest, Naive Bayes, Decision Tree, Machine Learning, Locky, WannaCry, Teslacrypt**

I. INTRODUCTION

In today's world, computers are an essential part of our lives. For instance, you can write a mail in a word processor, make changes to it, print multiple copies, send it to somebody via email halfway across the world in just a few clicks. All these tasks would have taken days to accomplish before, but now they are completed within moments. The fact of the matter is, whether it is a Small Business, or an Enterprise, they all rely on computer systems today. Pairing this with the rise in cloud computing, rather poor cloud security, and smart devices everywhere contributes to a multitude of threats.

Malware is a term used to define any piece of software built with the intention of harming the computer system, user or the data. These malwares were first introduced during the early 1970s to cause disruptions and as much damage as possible. Types of malwares include Viruses, Worms, Trojans and Spywares, furthermore in this research paper we will be focusing on detection of ransomwares.

Ransomware is a type of malign software, which when given access to a users computer system, will encrypt the data essentially rendering the system useless until the ransom amount is paid usually in the form of bitcoins. These softwares are able to set up their working environment in the background, also creating a backdoor to the system in the process to remain updated throughout the duration of attack. After the encryption is performed, the user is prompted with a ransom note (usually desktop wallpaper is set to the ransom note) with the details to the transaction ID and an intimidating note asking for money.

Ransomware's first cases date back to 2005 in Russia. Since then these types of malwares have spread across the world, advancing in method of implementation and targeting of their victims. Cryptolocker first surfaced in September 2013 and was advanced enough to target all versions of Windows. Victims would accidentally open these files and send them via email impersonating UPS, DHS etc. Once activated, the malware would get to work and prompt with a timer of 72 hours and a ransom.

In this paper, we have analysed some of the most popular ransomwares such as WannaCry, Locky, CryptoShield, Crysis, Win32.Blocker, Unlock26. These ransomware samples have been used from the ISOT Ransomware Dataset and contain trace files of the ransomwares.

We have used multiple machine learning algorithms and compared their results. The paper is divided into subsections as follows: Section Two will discuss related work on ransomware related research. The proposed methodology and implementation is discussed in the Fourth section. Section Five discusses the results observed. Finally, with Section Six we conclude our paper.

II. Related Work

There has been a surge in the use of machine learning approaches to detect and prevent ransomware attacks. A detailed study was made by Martina Jose Mary.M, Usharani.S , Thirugnanam.P[2] in their paper studied the detailed working of a ransomware attack and the different types of ransomware. They analyzed the features like CPU user usage, system usage, RAM usage, receive packet and byte, send packets, send bytes, receive packets, receive bytes, and netflows of the dataset. They used various machine learning algorithms such as KNN, Naïve Bayes, Random Forest, SGD, SVM, Logistic Regression, Bayesian Network to get desired output. They showed the comparison of all the accuracies and results as well. But every algorithm has its own advantages and disadvantages, so the best suited model which can be handled by the system is to be chosen. We improve on this study further by using different machine learning algorithms. We used the behaviour of the ransomware trace files to train and test the model.

A paper written by Ban Mohammed Khammas[3] is a static method of detection. They used a random forest classifier to detect and found out that a high accuracy can be achieved by changing the seed value to 1 and tree numbers 100. They also used Frequent Pattern Mining technique to directly extract the features from raw byte. This showed an increase in efficiency.

Next paper by Eduardo Berrueta, Daniel Morato, Eduardo Magaña, Mikel Izal[4]. They made an in depth survey that concentrates on various detection models. In comparison to the previous studies, they offer a survey on different ransomware families and propose a few detection algorithms.

In the paper by Cuneyt Gurcan, Yitao Li, Yulia R. Gel, Murat Kantarcioglu[5] , they used advanced data analytics techniques to find ransomware related transactions and bitcoin addresses. But using bitcoin technology has its own disadvantages of varying bitcoin transaction addresses.

In the paper by Li Chen, Chih-Yuan Yang, Anindya Paul, Ravi Sahita[6] , they have compared the flexibility and stability of Machine Learning algorithms for security. A case study was done on evaluation of resilience using the generative adversarial network. They emphasized on the necessity of improving machine learning based approaches before final deployment. We gained a lot of insight on the various strategies and using ML techniques. It helped us understand how ML models can be made more resilient and robust.

Lastly, a paper written by Edgar P. Torres P. and Sang Guun Yoo[7] helped us understand the current literature existing on the subject. Its survey consisted of research done in malware detection, prevention and recovery of the system after the attack. It helped us understand the existing work and further scope of research in this field from various papers in a summarized manner.

III. Proposed Methodology

A. Methodology

Ransomware has 5 stages in which it is able to attack the user's system. These 5 Stages are:

1. Distribution: Malware is made and distributed through the internet.
2. Infection: Malware enters the system through the help of social engineering attack or other malwares such as trojans.
3. Staging: In this stage, ransomware application tries to distribute itself into multiple folders or tries to create a working environment in the system.
4. Scanning: In this stage, the ransomware scans for files that may be of importance to the user.
5. Encryption: The ransomware after scanning the files has what it requires and starts the file encryption process
6. Payday: A Ransom note appears on the users screen usually in the form of text or wallpaper.

From the above, we can say, that Stage 3 and 4 are essential stages for the successful execution of any ransomware. Hence, by taking countermeasures before or during these stages to prevent it from causing major damages to the user's system.

Using an efficient Machine Learning model will help in analysing and detecting the movements of the malware much earlier. In this research, we will be focusing on the optimum Machine Learning model that could be used for this application. We will be concentrating mainly on four algorithms.

B. Machine Learning Classifiers Used

- ***KNN (K - Nearest Neighbor):*** It is one of the easiest algorithms used for classification and regression. It is a non parametric method, which is also known as the lazy learning model with local approximation. It uses data points that are most similar as a base to classify data points. The working of KNN can be briefly explained as:
    - Select K as the number of neighbors
    - Calculation of Euclidean distance of K number of neighbors
    - Select nearest neighbors as the the calculations

- Among these, calculate the number of data points in each categories
- Assign the newer data to the categories

$$\text{Euclidean Distance between } A_1 \text{ and } B_2 = \sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$$

where, A1 and B2 are data points and (X1,Y1) and (X2,Y2) are their coordinates respectively.

- ***Decision Tree:*** This type of algorithm uses supervised machine learning in which the data is split based on a parameter. Decision Nodes and Leaves are the two entities of a tree. Leaves are the final outcomes, whereas the decision nodes are where the data is split.

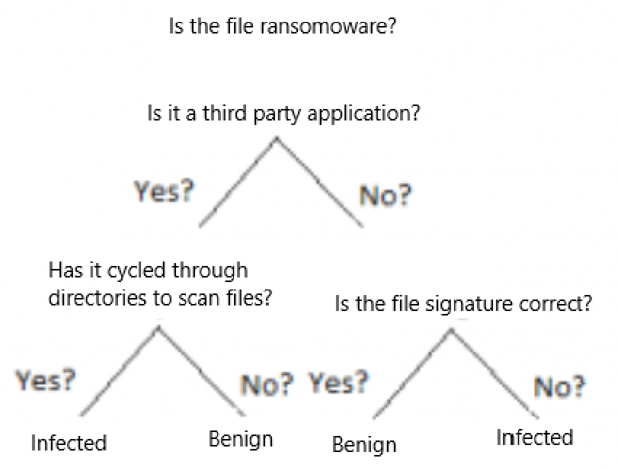

Is the file ransomoware?
Is it a third party application?
Yes?
No?
Has it cycled through directories to scan files?
Is the file signature correct?
Yes?
No?
Yes?
No?
Infected
Benign
Benign
Infected

FIG. 1 - Decision Tree representation

In the above example, "Is it a third party application?" and "Benign/Infected" are Decision nodes and leaves respectively. The working of Decision tree can be briefly explained as:

- Create a root node.
- Return 'Positive' value for leaf node, if all samples are positive.
- Return 'Negative' value for leaf node, if all samples are negative.
- Entropy for the Current State H(S) must be calculated
- Entropy must be calculated, for each attribute corresponding to its attribute 'x' which is represented as H(S, x).
- Attribute having the largest value of IG(S, x) must be selected.
- Attribute offering the greatest IG must be removed from the rest of the attributes.
- This must be repeated until we have no more attributes, or when the decision tree consists of only leaf nodes

$$H(S) = \sum_{x \in X} p(x) \log_2 \frac{1}{p(x)}$$

Where,

$$IG(S, A) = H(S) - H(S, A)$$

Alternatively,

$$IG(S, A) = H(S) - \sum_{i=0}^{n} P(x) * H(x)$$

Information Gain (IG) is represented by IG(S,A), where in Set, change of entropy is denoted by S on a particular attribute A.

- ***Random Forest:*** It is an easy-to-use algorithm that gives good results even without hyper-parameter tuning. It is optimal for both regression and classification tasks because of its flexible and diverse nature. It builds a “forest” that is a collection of decision trees trained using a method known as ”bagging”. It combines learning models that increases the results. The working of Random Forest can be briefly explained as:

  - First random samples from the dataset are selected.
  - A decision tree for every sample will be constructed.
  - Prediction results from every decision tree are noted.
  - Voting will be performed for every result that is predicted.
  - Finally, we will select the most voted result as the final prediction result.

$$RFfi_i = \frac{\sum_{j \in all\ trees} normfi_{ij}}{T}$$

RFfi sub(i)= Significance of feature ‘i’ by calculating from all the trees in the model.
normfi sub(ij)= Significance of the normalized feature for ‘i’ in tree ‘j’
T = Number of total trees.

- ***Gaussian Naïve Bayes:*** It is a group of supervised algorithms that use Bayes theorem for classification. It is an easy classification technique with high functionality. Gaussian Naïve Bayes uses Gaussian Normal Distribution. It is used for continuous data distribution. After calculating the probabilities for input values for each class using a frequency, we calculate the mean and standard deviation from the training data.

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

- ○ We must assume the data given to us is defined by gaussian distribution having no independant dimensions nor co-variance.
- ○ We can make the model fit by finding the mean and standard deviation of the points within each label.
- ○ At every data point, the distance between z-score distance and each class-mean is calculated, i.e, distance from class mean divided by the standard deviation.

C. *System Architecture*

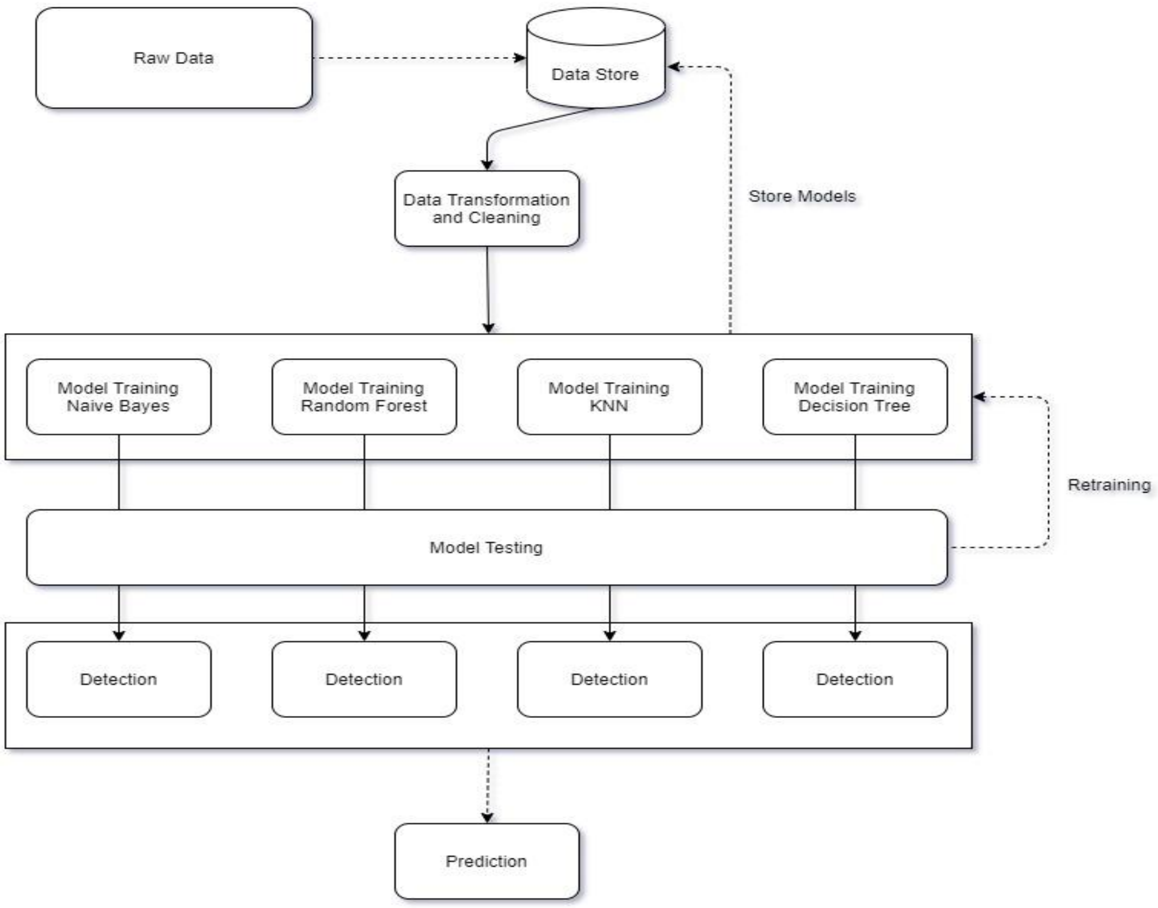


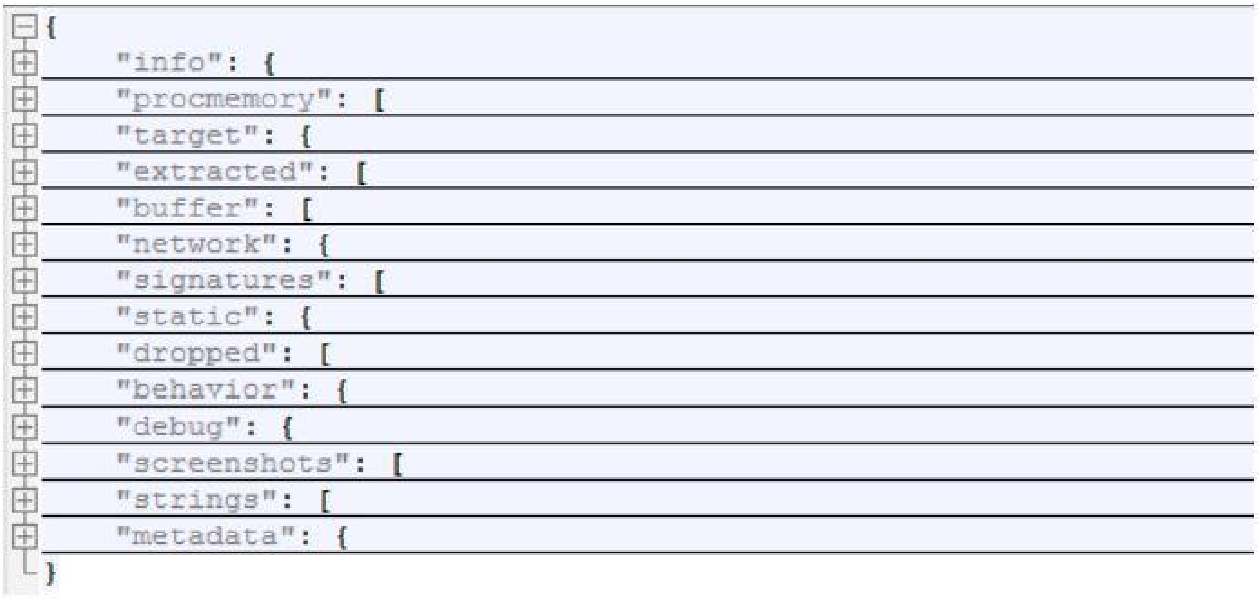
FIG. 2 - System Architecture

- Raw Data : This consists of unorganized raw data. It contains trace files from existing ransomwares about their behavioural data from the ISOT Ransomware Detection Dataset. The file format is '.json' and must be converted to '.csv' format to make it compatible with the models.

- Data Store : This module converts the '.json' files to '.csv' files and stores them for further use.
- Data Transformation and Cleaning: This module is used to clean the data by removing NaN values, converting the data using label encoding to convert the string values to the numeric form readable by the machine learning algorithms. This module also splits the data into the training, and validation set at ratio 0.25 (75:25 ratio).
- Model Training : This module firstly creates a cross validation set that is utilized by the models for training. We use k-fold value as 3 to better estimate the machine learning model skill on unseen data. It also contains the four Machine Learning Models (KNN, Decision Tree, Naive Bayes, Random Forest) that will process the data fed in from the training dataset and get trained. These models are tuned accordingly using Hyperparameters for each model and optimizing it to give the best accuracy possible.
- Validation : This module will use the trained model on the testing dataset, which will allow us to validate our results. Along with Model accuracy, it will also give us the precision, recall, f1 score and the confusion matrix for the respective models.

D. *Dataset*

We used the ISOT Ransomware Detection Dataset from the ISOT Research Lab, University of Victoria. It is a behaviour data of a collection of ransomware and benign samples. They were obtained from Virustotal under academic license along with several samples from anti-malware companies. The dataset consists of a total of 669 ransomware samples from ransomware families that are most widely used. The size of the entire dataset on disk is 428 GB. Apart from the ransomware samples, the dataset also includes 103 benign most used windows applications.



```
{
    "info": {
    "procmemory": [
    "target": {
    "extracted": [
    "buffer": [
    "network": {
    "signatures": [
    "static": {
    "dropped": [
    "behavior": {
    "debug": {
    "screenshots": [
    "strings": [
    "metadata": {
}
```

FIG. 3 - Sample Report Top Level Structure

A malware analysis sandbox called the cuckoo was installed in the host machine. It is a free open-source software tool that automates the analysis of malicious files. The dataset contains information about the analysis task, duration analysis, various memory regions, established network connections, processes created by sample, system API calls during the initial analysis, arguments, return values, strings extracted from the binary file of the analysed sample, different operations on file system and Windows registry.

E. Evaluation

- Classification Accuracy - It is the correct number of predictions divided by the number of total input samples. It is given as-

$$Accuracy = \frac{\text{Number of Correct Predictions}}{\text{Total number of predictions Made}}$$

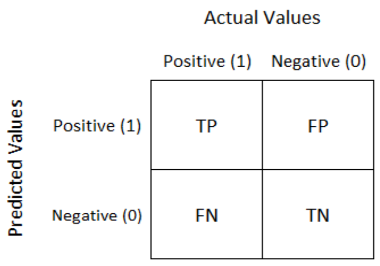


FIG. 4 - Confusion Matrix Representation

- Confusion Matrix- The output is given in the matrix form and it describes the models complete performance.
  Where,
  TP: True Positive   TN: True Negative
  FP: False Positive   FN: False Negative

- F1 score- Test's accuracy can be measured by it. It is also the Harmonic Mean between recall and precision. The score ranges from [0,1]. It tells how precise and robust your model is. Mathematically, it is given as -

$$F1 = 2 * \frac{1}{\left(\frac{1}{precision}\right) + \left(\frac{1}{recall}\right)}$$

  F1 Score finds the equilibrium between Recall and Precision.

- Precision: It is the ratio between the number of true positive results to the number of all positive results. It can be expressed as-

$$Precision = \frac{TP}{(TP + FP)}$$

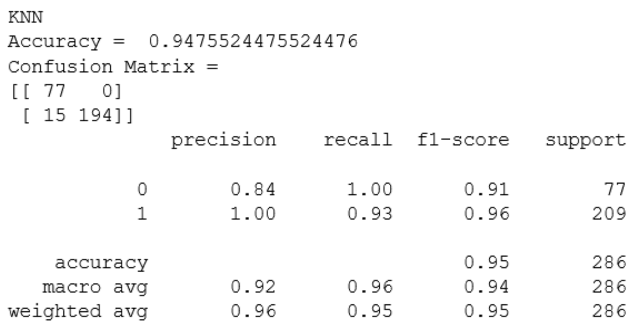
- Recall: It is the ratio between the number of true positive results to the total relevant samples. In mathematical form it is given as-

$$\text{Recall} = \frac{TP}{(TP + FN)}$$

IV. Results

The following results were observed:

- **KNN** achieved an accuracy of 94.7% but as it is a lazy learning algorithm its execution cost will tend to be greater on bigger datasets.



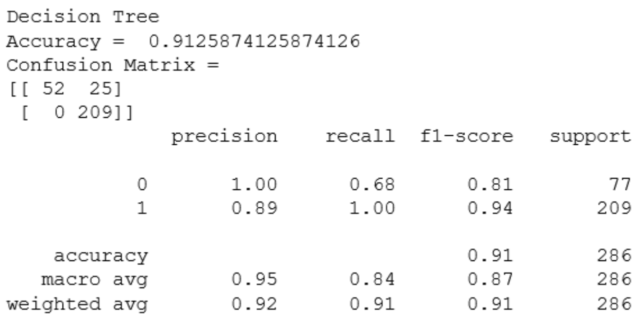
```
KNN
Accuracy =  0.9475524475524476
Confusion Matrix =
[[ 77   0]
 [ 15 194]]
              precision    recall  f1-score   support

           0       0.84      1.00      0.91        77
           1       1.00      0.93      0.96       209

    accuracy                           0.95       286
   macro avg       0.92      0.96      0.94       286
weighted avg       0.96      0.95      0.95       286
```

FIG. 5 - KNN Results

- **Decision Tree** achieved an accuracy of 91.1%



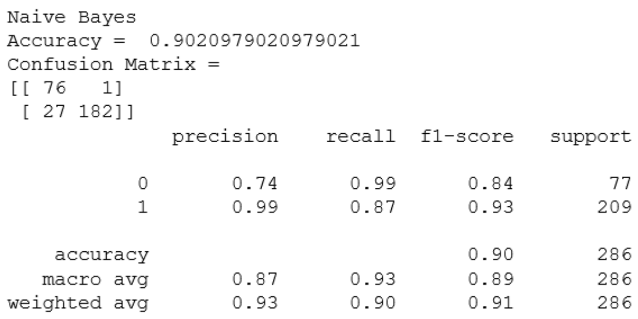
```
Decision Tree
Accuracy =  0.9125874125874126
Confusion Matrix =
[[ 52  25]
 [  0 209]]
              precision    recall  f1-score   support

           0       1.00      0.68      0.81        77
           1       0.89      1.00      0.94       209

    accuracy                           0.91       286
   macro avg       0.95      0.84      0.87       286
weighted avg       0.92      0.91      0.91       286
```

FIG. 6 - Decision Tree Results

- **Naive Bayes** achieved an accuracy of 90.2%



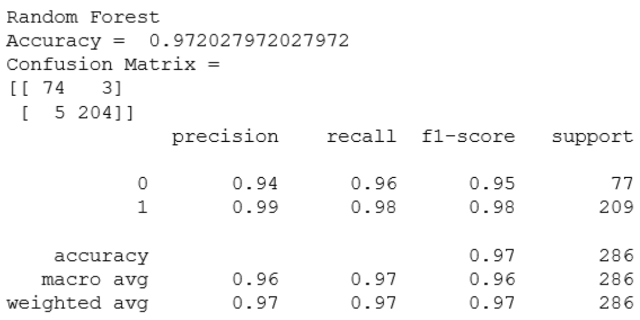
```
Naive Bayes
Accuracy =  0.9020979020979021
Confusion Matrix =
[[ 76   1]
 [ 27 182]]
              precision    recall  f1-score   support

           0       0.74      0.99      0.84        77
           1       0.99      0.87      0.93       209

    accuracy                           0.90       286
   macro avg       0.87      0.93      0.89       286
weighted avg       0.93      0.90      0.91       286
```

FIG. 7 - Naive Bayes Results

- **Random Forest** was able to achieve the highest accuracy of 97.2%. This was tuned with a higher number of trees to improve the accuracy and give more stable results, hence, impacted the performance slightly taking a longer duration (1.28s)



```
Random Forest
Accuracy =  0.972027972027972
Confusion Matrix =
[[ 74   3]
 [  5 204]]
              precision    recall  f1-score   support

           0       0.94      0.96      0.95        77
           1       0.99      0.98      0.98       209

    accuracy                           0.97       286
   macro avg       0.96      0.97      0.96       286
weighted avg       0.97      0.97      0.97       286
```

FIG. 8 - Random Forest Results

.

V. Conclusion and Future Work

Over the years, researchers have helped develop various strategies to aid in detecting and preventing malware attacks. Signature based detection is one of the most popular strategies used, but is challenged by newer and advanced malwares being churned out by hackers. Hence, an efficient and accurate dynamic based machine learning approach is required.

As we can conclude, from the results shown in the previous section:

- Random Forest is able to perform with higher accuracy compared to the other three models (Naive Bayes, KNN, Decision Tree). This model is able to distinguish between benign and infected files efficiently using the traces provided during the training and validation process. Though, the Random Forest model has an expensive cost of estimation trees, which is slightly heavier on the system, but provides a more stronger and stable accuracy.
- Another good alternative to the random forest model is the KNN model, as it has also performed exceptionally well. Though, this KNN model is more suitable with smaller datasets due to high real time execution cost but allows for on the fly addition of data without affecting the accuracy of the system.

The future scope of this research is to implement the system in real time and test it thoroughly. This system can be further improved on, if paired with a signature based static method and will help detect malicious files earlier if code was not modified to bypass the signature based approach, but further research is required on that.

VI. Reference

[1] Brijesh Jethva, Issa Traoré, Asem Ghaleb, Karim Ganame, Sherif Ahmed, **“Multilayer Ransomware Detection using Grouped Registry Key Operations, File Entropy and File Signature Monitoring”** Journal of Computer Security, **Volume 28, Number 3, 2020, pages 337-373.**

[2] Martina Jose Mary.M1 , Usharani.S², Thirugnanam.P³, **“Detection and Deterrence of Ransomware using Machine Learning Techniques: A survey”** , Journal of Information and Computational Science, **Volume 10 Issue 1 – 2020**.

[3] Ban Mohammed Khammas, **“Ransomware Detection using Random Forest Technique”**, ICT Express, **volume 6, issue 4, December 2020, Pages- 325-331.**

[4] Eduardo Berrueta, Daniel Morato, Eduardo Magaña, Mikel Izal, **“A Survey on Detection Techniques for Cryptographic Ransomware”** , IEEE Access,Volume 7, pages-144925 – 144944, 07 October 2019.

[5] Cuneyt Gurcan, Yitao Li, Yulia R. Gel, Murat Kantarcioglu, **“ BitcoinHeist: Topological Data Analysis for Ransomware Detection on the Bitcoin Blockchain ”**, arXiv:1906.07852v1 **[cs.CR]** , 19 june 2019.

[6] Li Chen, Chih-Yuan Yang, Anindya Paul, Ravi Sahita, **” Towards Resilient Machine Learning for Ransomware Detection”** , arXiv:1812.09400v2 **[cs.LG]** ,2018.

[7] Edgar P. Torres P. and Sang Guun Yoo, **“ Detecting and Neutralizing Encrypting Ransomware Attacks by Using Machine-Learning Techniques: A Literature Review ”**, International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 18 (2017) pp. 7902-7911.

[8] Samah Alsoghyer, Iman Almomani, **“Ransomware Detection for Android Applications”**, *Electronics* **2019**, *8*(8), 868; **https://doi.org/10.3390/electronics8080868**

[9] S. H. Kok , Azween Abdullah , N. Z. JhanJhi, Mahadevan Supramaniam , **“Prevention of Crypto-Ransomware Using a Pre-Encryption Detection Algorithm”**, *Computers* **2019**, *8*(4), 79; **https://doi.org/10.3390/computers8040079**

[10] A Deep Neural Network Strategy to Distinguish and Avoid Cyber Attack, Siddhant Agarwal, Abhay Tyagi ,G. Usha, Artificial Intelligence and Evolutionary Computations in Engineering Systems pp 673-681, Part of the Advances in Intelligent Systems and Computing book series (AISC, volume 1056).

[11] J. Jeyasudha and G. Usha, **"Detection of Spammers in the Reconnaissance Phase by machine learning techniques,"** 2018 3rd International Conference on Inventive Computation Technologies (ICICT), Coimbatore, India, 2018, pp. 216-220, doi: 10.1109/ICICT43934.2018.9034457.

[12] E. S. Domi and G. Usha, **"TSDLA: Algorithm for Location Privacy in Clustered LB-MCS network,"** *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*, Tirunelveli, India, 2020, pp. 68-74, doi: 10.1109/ICSSIT48917.2020.9214260.