Malware Detection Using ML

Megha Nayanshi[1], Km Gunjan[2], Sony Saini[3], Dr. Ankit Kumar[4]
[1,2,3]*B. Tech, CSE, 3rd Year, Galgotias University*
[4]*Department of Computer Science & Engineering, Galgotias University*

***Abstract* - Malware is a set of different programs or any software which has a primary aim of trying to cause the damage/harm to a computer, computer network, client, or a server. Malware starts damaging the system after it is introduced into the target's computer and it can take any form such as scripts, so-called "active content" (Microsoft Windows), code which can directly execute and/or other forms of data. There are some malwares that are mostly referred as computer viruses, Trojan horses, scareware, ransomware, adware, spyware, and worms etc.**

To investigate that how we can implement machine learning technique on malware detection for the purpose of detecting any unknown malware, for developing a software that implements machine learning for detecting the unfamiliar malware & to validate that. Malware detection that uses machine learning will be capable to obtain high accuracy rate with the less false positive rate.

1:INTRODUCTION

1.1 PURPOSE

The most baleful problem of the different malware is that can attack and harm your system. In present time malware is one of the greatest safety fears on the computer/internet. These days, malware is the primary cause for maximum Cyberspace complications for instance junk mails and dos attacks. The PC that are undermined with malware are frequently arranged together to frame botnets, and numerous assaults are propelled utilizing these malignant, assailant controlled systems.

Various new methods and procedures are used to detect and prevent any further damages raised by malware.

1.2 OBJECTIVE

Implementing machine learning for malware detection for the purpose of detecting malware files. To develop the malware detection algorithm that implements machine learning to detect malware. To validate that machine learning will detect malware with high rate of accuracy with a lowfalse positive rate.

2: LITERATURE REVIEWS

2.1: EXISTING SYSTEM

Traditional security software or algorithms practices old-virus scanner for detecting faulty/malicious program/code, they uses “signature-based detection” to detect malware. But with malware that have become polymorphic and metamorphic the traditional used detection method i.e, signature-based detection method by antivirus are no longer affective. In current anti-malware software, there are 2 major tasks that are approved by the malwar analysing procedure, that is malware recognition and malware sorting.

2.2: PROPOSED SYSTEM

We are focusing on malware discovery. Our focus is to distinguish “malware in the system”. 2 commonly used malware detection are static analysis and dynamic analysis. Aimed for efficient and effective recognition, the necessity of extraction of feature for “malware detection” is vital. After extracting feature from the categories, we have to combine all category into a feature vector for classifier to run on them. For building a learning algorithm, features are haul out from the categories will experience arrangement with using different methods like, KNN, Random Forest and many others, but Random Forest is the preferred one as the data contains some noise, and overfitting risks in the extracted feature. To get outcome of the machine learning based model is to test different dataset with tag to produce a graph which specify the rate of detection and false positive rate. To find the best outcome, repeat the process using many other classifications and create learning model to test on the same dataset. The best outcome is gained when the graph has maximum detection rate and lowest false positive rate.

3: SYSTEM REQUIREMENTS

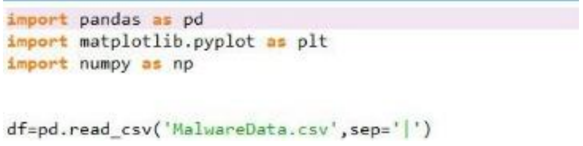
3.1: Hardware interface
Device: Laptop, Smart Phones or Desktop Computer
Processor: CORE i3 (3rd Gen minimum) and above
RAM: 4GB(minimum) and above
Hard disk: 100 GB (minimum) and above

3.2: Software Requirements
Operating System: Windows, Linux – Ubuntu
Platforms: Jupyter, Spyder, Google Collab, Anaconda prompt, Virtual Box Languages: Python
Web browsers: Chrome, Firefox

4: OVERALL DESCRIPTION

4.1: Project Perspective
The general thinking behind machine learning is to make the algorithm such that it can learn by itself the best parameters from data such that it makes good predictions. There are many applications used for malware detection, but here we will be using machine learning to classify files between malicious and legitimate files.

4.2: Project function
4.2.1: Initialization
The dataset is loaded from the file and is saved in memory.

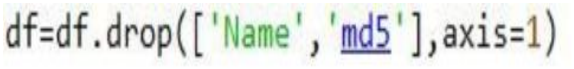


```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np


df=pd.read_csv('MalwareData.csv',sep='|')
```

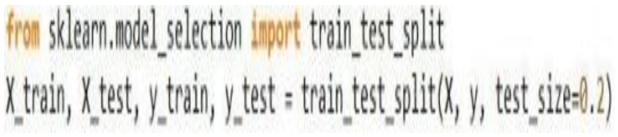
4.2.2 Feature Extraction
Since every feature is in numeric form, therefore there is no need for this particular step.

4.2.3 Feature Selection
Since the Column "Name" only signifies the name of the file and Column "md5" is a hash function.
So these values won't affect the training model and therefore we can drop them.



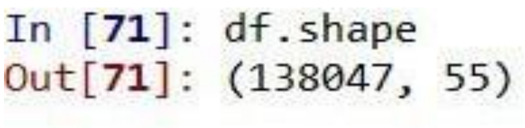
```
df=df.drop(['Name','md5'],axis=1)
```

4.2.4 Splitting the data



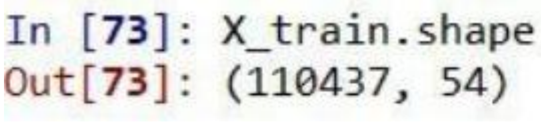
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

Original data:



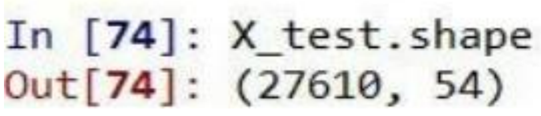
```
In [71]: df.shape
Out[71]: (138047, 55)
```

Training data :



```
In [73]: X_train.shape
Out[73]: (110437, 54)
```

Test data :



```
In [74]: X_test.shape
Out[74]: (27610, 54)
```

.

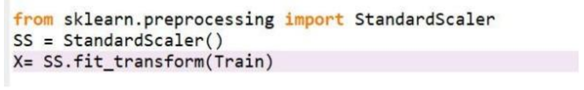
4.2.5: Data Pre-processing
1. Standard Scalar
Standard Scalar is a technique used to standardize features by scaling to variance and removing mean.
The standard score of a sample 'x' is generated by the formula:

$$z = (x - u) / s$$

where u =mean

and s = standard deviation

Standardizing the dataset is a basic requirement for many learning models and algorithms, as they might act badly if the individual features does not belong to same scale.



```
from sklearn.preprocessing import StandardScaler
SS = StandardScaler()
X= SS.fit_transform(Train)
```

4.2.5: Applying Classification Models
1. Logistic Regression (LR)
Logistic Regression (LR) is type of a supervised learning algorithm & it is used for classification. In Logistic Regression Algorithm, the probabilities which are relating to the total possible outcomes of a single event are modelled using this algorithm.

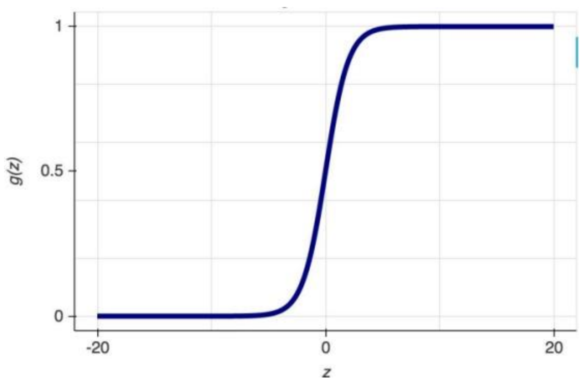


Fig. 1 Logistic Function

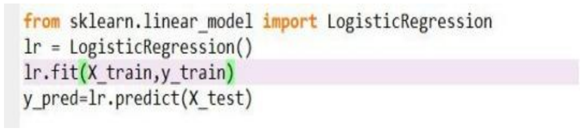
The function that is used as logistic – function is known as sigmoid function.

$$y = \frac{1}{1+e^{-z}}$$

Advantages:
Logistic Regression is made for classification and is generally helpful for knowing how different autonomous factors influences on a solitary result or dependent variable.

Disadvantages:
Logistic Regression possibly works when the anticipated variable is double, presumes that all indicators are independent from one another, and information is liberated from Nan esteems/values.

Applying Logistic Regression model:



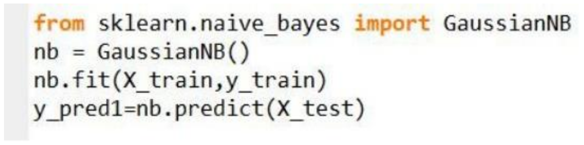
```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(X_train,y_train)
y_pred=lr.predict(X_test)
```

2. Naïve Bayes
Naïve Bayes is based on the Bayes theorem which makes the assumption of independence between every pair of features. Naïve Bayes classifiers has many real-world applications such as classifying document and spam filtering.

Advantages:
It requires small training dataset to estimate the vital factors/parameters. It is very fast as compared to other methods.

Disadvantages:
It is also a bad estimator as it trains the model for small dataset.

Applying Naive Bayes model:



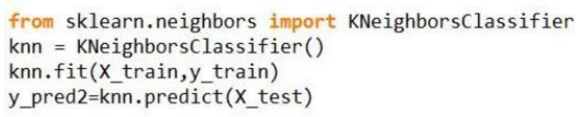
```
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(X_train,y_train)
y_pred1=nb.predict(X_test)
```

3. K-Nearest Neighbor
KNN based classifiers are kind of lazy learners as they do not try to build a general internal model but stores the instances of the training data. Classification is generated from a simple majority vote of the k nearest neighbours for each point".

Advantages:
This calculation is very simple to actualize, viable for huge measure of training data. and powerful to data containing anomalies and commotion.

Disadvantages:
It requires to calculate the value of K and the cost of computation is quite high since it computes the distance of each instance to every training samples.

Applying K-Nearest Neighbour model:



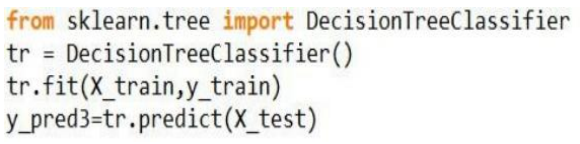
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(X_train,y_train)
y_pred2=knn.predict(X_test)
```

4. Decision Tree (DT)
DT introduces a set of rule's which helps in classifying the data, given the data with attributes and its classes.

Advantages:
DT are easy visualize, understand and implement as it requires less preparation, & it handle's categorical as well as numerical data.

Disadvantages:
DT makes complex tree structure that doesn't sum up well, and decision trees can be temperamental as little varieties in the information could bring about creating a totally unique tree.

Applying Decision Tree model:

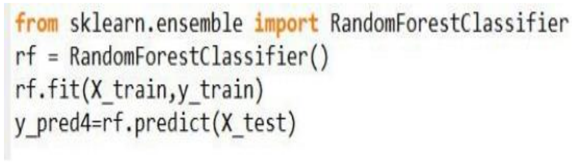


```
from sklearn.tree import DecisionTreeClassifier
tr = DecisionTreeClassifier()
tr.fit(X_train,y_train)
y_pred3=tr.predict(X_test)
```

5. Random Forest
Random Forest Classifier fits a number of Decision Trees on different sub samples of datasets and uses averages to improve the prediction accuracy of the model and helps in controls over-fitting problem. The size of "sub sample remains the same as the original input sample size given but these samples are drawn with replacement".

Advantages:
Decrease in the over-fitting issue, accuracy of Random Forest is more prominent than the Decision Tree much of the time.

Disadvantages:
Real-time predictions are slow, usage is troublesome/hard and the algorithm is very perplexing/complex.
Applying Random Forest model:

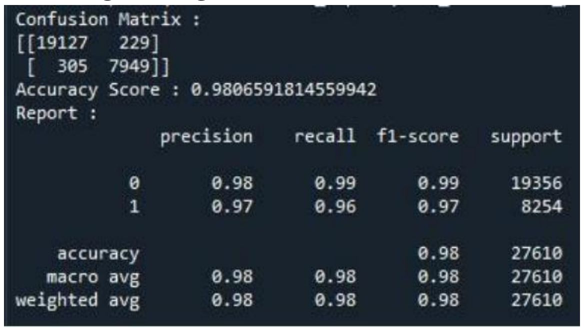


```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(X_train,y_train)
y_pred4=rf.predict(X_test)
```

5. CONCLUSION

5.1: Conclusion
The ideal component extraction and representative techniques were chosen and the chosen machine learning algorithms were applied and assessed/evaluated. The results of the Classification models used above is shown below.

5.1.1: Logistic Regression:

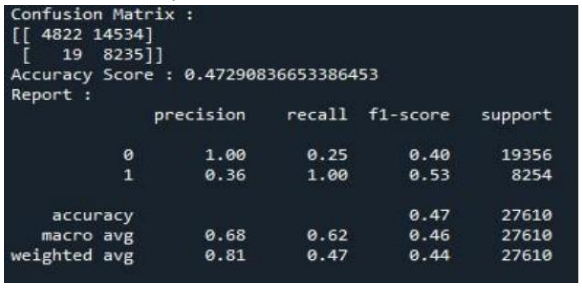


```
Confusion Matrix :
[[19127   229]
 [  305  7949]]
Accuracy Score : 0.9806591814559942
Report :
              precision    recall  f1-score   support

           0       0.98      0.99      0.99     19356
           1       0.97      0.96      0.97      8254

    accuracy                           0.98     27610
   macro avg       0.98      0.98      0.98     27610
weighted avg       0.98      0.98      0.98     27610
```

5.1.2: Naïve Bayes:

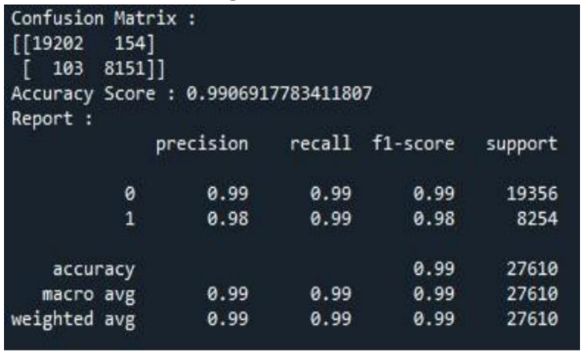


```
Confusion Matrix :
[[ 4822 14534]
 [   19  8235]]
Accuracy Score : 0.47290836653386453
Report :
              precision    recall  f1-score   support

           0       1.00      0.25      0.40     19356
           1       0.36      1.00      0.53      8254

    accuracy                           0.47     27610
   macro avg       0.68      0.62      0.46     27610
weighted avg       0.81      0.47      0.44     27610
```

5.1.3: K-Nearest Neighbour:

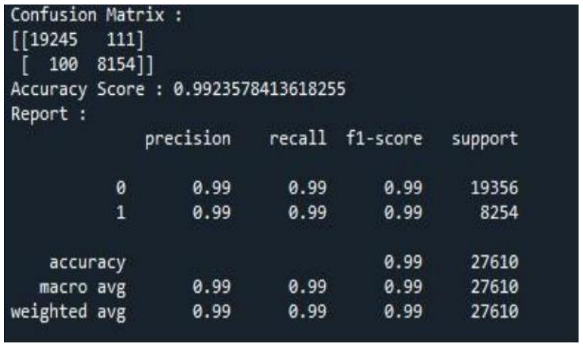


```
Confusion Matrix :
[[19202   154]
 [  103  8151]]
Accuracy Score : 0.9906917783411807
Report :
              precision    recall  f1-score   support

           0       0.99      0.99      0.99     19356
           1       0.98      0.99      0.98      8254

    accuracy                           0.99     27610
   macro avg       0.99      0.99      0.99     27610
weighted avg       0.99      0.99      0.99     27610
```

5.1.4: Decision Tree (DT):

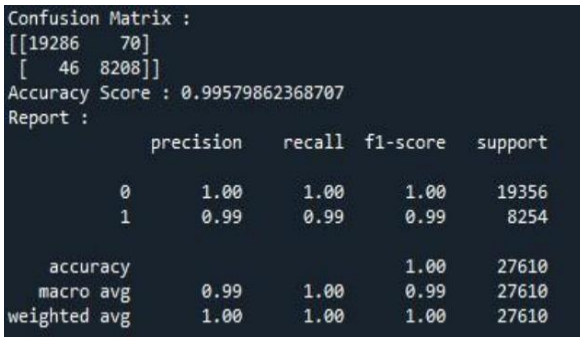


```
Confusion Matrix :
[[19245   111]
 [  100  8154]]
Accuracy Score : 0.9923578413618255
Report :
              precision    recall  f1-score   support

           0       0.99      0.99      0.99     19356
           1       0.99      0.99      0.99      8254

    accuracy                           0.99     27610
   macro avg       0.99      0.99      0.99     27610
weighted avg       0.99      0.99      0.99     27610
```

5.1.5: Random Forest:



```
Confusion Matrix :
[[19286    70]
 [   46  8208]]
Accuracy Score : 0.99579862368707
Report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     19356
           1       0.99      0.99      0.99      8254

    accuracy                           1.00     27610
   macro avg       0.99      1.00      0.99     27610
weighted avg       1.00      1.00      1.00     27610
```

5.1.6: Comparision of Models:
Table 1: Comparison of Models.

Algorithm	Precision	Recall	Accuracy
Logistic Regression	0.97	0.97	98.06
Naïve Bayes	0.36	0.53	47.29
K-Nearest Neighbors	0.98	0.98	99.06
Decision Tree	0.99	0.99	99.23
Random forest	0.99	0.99	99.57

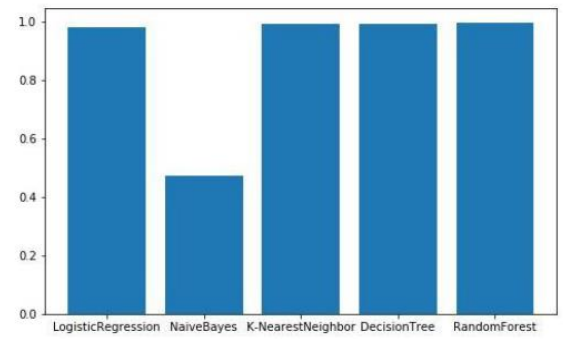


Fig. 2 Accuracy of Models

Table 2: Comparison of ROC Curve

Model	Sensitivity For ROC Cureve
Logistic Regression	0.976
Naive Bayes	0.623
K-Nearest Neighbour	0.990
Decision Tree	0.991
Random Forest	0.995

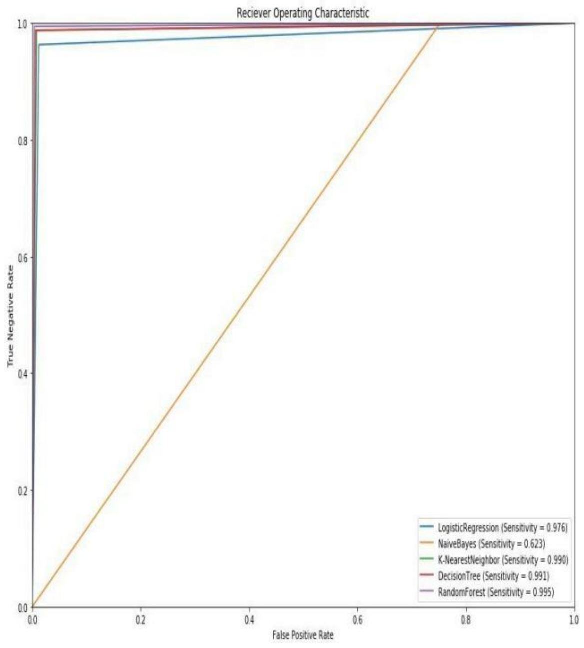


Fig. 3 ROC Curve

From the results above its clear that all models except Naïve Bayes are showing the best results on the given data. While Naïve Bayes shows poor result because of its probability function.

5.2: Future Scope

5.2.1: Advanced Preprocessing

Till now the preprocessing used is of basic ground level. Further advanced preprocessing techniques like Normalization, Encoding and Dimensionality Reduction (e.g. PCA) can be applied on the dataset.

5.2.2: Use a wider dataset

Currently we have used the data which is in csv file format. Now will be using data which is in image format and in ASM & BYTES file format.

Additionally, the dataset that was utilized in this investigation is tight, however it covers a large portion of the malware types that exist in the cutting-edge world, however it doesn't cover every conceivable sort. Gathering malware datasets is a tedious undertaking which requires a ton of exertion. For increasingly precise aftereffects of the models, it is encouraged to test the models for all the potential kinds of malwares like spyware, adware, rootkits, backdoor, banking malware, etc. Additionally, comprehend that model may have the option to anticipate the examples of the families that has been seen before. At the end of the day, in a true application, the most extreme measure of potential families ought to be utilized before the dispatch of the venture for true conditions.

REFERENCES

WEBSITES:

[1] https://www.blackhat.com/docs/us-17/thursday/us-17-Anderson-Bot-Vs-Bot-Evadi ng-Machine-Learning-Malware-Detection-wp.pdf

[2] https://en.wikipedia.org/wiki/Malware https://www.kaggle.com/

[3] https://www.infosecurity-magazine.com/opinions/malware-detection-signature4.

[4] https://www.kaggle.com/

[5] https://towardsdatascience.com/