



**VIT<sup>®</sup>**

**Vellore Institute of Technology**  
(Deemed to be University under section 3 of UGC Act, 1956)

# **MALWARE DETECTION USING MACHINE LEARNING - DT, SVM, RF, XGBOOST**

## **Team Members:**

**19BCE0907 (Kumar Shubham)**

**19BDS0006 (Rashi Maheshwari)**

**19MIC0123 (Santhosh Kumar)**

**Course Code: BCI4003**

**Malware Analysis**

**Slot: G1 Slot**

**Professor: Vijaya Kumar K**

## **Abstract:**

Nowadays, attackers can design malware that continuously changes its recognizable feature to fool detection techniques that use typical signature-based methods. That is why the need for machine learning based detection arises. Our project aims to present the functionality and accuracy of various machine learning algorithms to detect malware. The machine learning algorithms that we will be focusing on are Decision Tree, SVM, Random Forest and XGBoost. The implementation would be carried out in python using Google Colab notebook. Colab notebooks are Jupyter notebooks that are hosted by Colab.

## **Introduction:**

Malware is any software intentionally designed to cause damage to a computer, server, client, or computer network. A wide variety of malware types exist, including computer viruses, worms, trojan horses, ransomware, spyware, adware, rogue software, wiper and scareware. Since the number of malicious software is rapidly increasing, antivirus companies are continuously looking for a technique that is the most effective in detecting malware. Signature based detection is the most popular method used by antivirus companies. However, the traditional malware detection strategies are not capable to notify the unknown malwares and only identify variants of malware that have been previously identified.[3]

Nowadays, attackers can design malware that continuously changes its recognizable feature to fool detection techniques that use typical signature-based methods. That is why the need for machine learning based detection arises. Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention. There are three types of machine learning algorithms: supervised learning, unsupervised learning and reinforcement learning. Machine learning algorithms can be used to identify the malware. The machine learning algorithms that we will be focusing on are Decision Tree, SVM, Random Forest and XGBoost.

A decision tree solves problems by automatically generating interrogation while training samples. For each node of the tree, a question is employed to decide whether the sample is a malware or a benign ware.[4]

The support vector machine classifier will also draw a hyperplane that splits malware from benign wear in the training dataset. The decision of being clean or suspicious depends on its location compared to the hyperplane.[4]

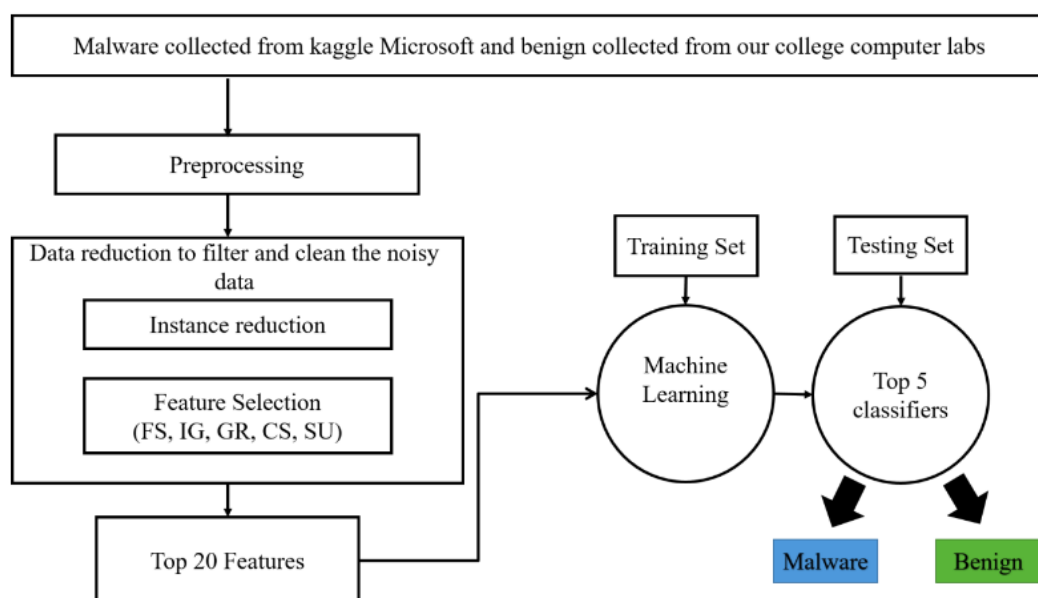
The random forest algorithm combines multiple decision trees where each tree is trained using different questions. Each tree was trained using a randomly chosen partial set of samples and the features of each set are randomly selected. The detection of a sample is

performed on every tree and the algorithm decides on the maliciousness of the binary based on the response of the majority of the trees.[4]

The XGBoost algorithm has the advantages of high speed, high precision and low resource consumption. It can be parallelized to deal with big data. It combines feature importance and expert evaluation methods to calculate course relevance. It not only guarantees objective correctness, but also takes subjective factors and human prior knowledge into consideration, which can significantly improve the robustness of the model.[7]

## Related Work:

### Survey 1:



In this paper, an approach based on opcodes occurrence to improve malware detection accuracy of the unknown advanced malware is discussed.

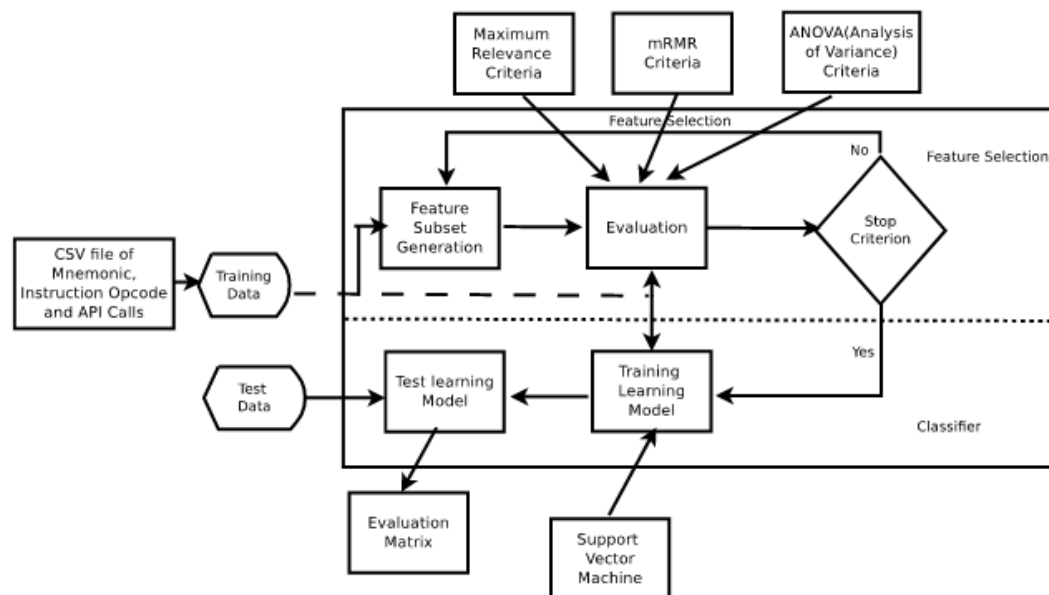
Code obfuscation technique is a challenge for signature based techniques used by advanced malware to evade anti-malware tools.

Proposed approach uses Fisher Score method for the feature selection and five classifiers used to uncover the unknown malware. In proposed approach Random forest, LMT, J48 Graft, and NBT detect malware with 100% accuracy which is better than the accuracy (99.8%) reported by Ahmadi et al.[1]

## Survey 2:

Title: Detection of Malicious Software by Analyzing Distinct Artifacts Using Machine Learning and Deep Learning Algorithms

Paper Discusses about various approaches and their respective drawbacks in implementation.



They calculate the relevance value of an attribute using the formula  $V(x) = I(h, x)$ , where  $h$  is the target variable or class,  $I(h, x)$  is the mutual information between class and feature  $x$ . [2]

## Survey 3:

In the current research, Machine Learning (ML) algorithm techniques became more popular to the researchers to analyse malware detection. In this paper, researchers proposed a defence system that used static analysis to extract the features based on PE information by comparing three different classifiers based on machine learning method algorithm techniques, K-Nearest Neighbor (KNN), Decision tree(DT) and Super Vector Machine (SVM), compares and selects them based on the high accuracy malware detection. The result indicates that DecisionTree algorithm is the best detection accuracy compared to other classifiers with 99% and 0.021% False Positive Rate (FPR) on a relatively small dataset. From this paper, it is clear that by using static analysis based on PE information and selecting the relevant features of the data can also give the best detection accuracy and can accurately represent malware. [3]

#### Survey 4:

Several ML algorithms are incorporated depending on their relevance. This paper focuses on various algorithms including Random Forest, Logistic Regression, Naive Bayes, Support Vector Machine (SVM), K-Nearest neighbors (KNN) and neural networks. The evaluation of the algorithms considered multiple malware features including PE headers, instructions, calls, strings, compression and the Import Address Table. The implementation was based on Python and sklearn. The models were trained on datasets from (Saxe and Sanders, 2018). The evaluation between these algorithms includes the Receiver Operating Characteristic Curve (ROC Curve), the detection time and the limitation of each algorithm.[4]

#### Survey 5:

Attacking machine learning infosec models

Several recent works have addressed attacking machine learning models in information security. In this they have categorised these methods into three coarse bins, which are graphically portrayed in Figure.

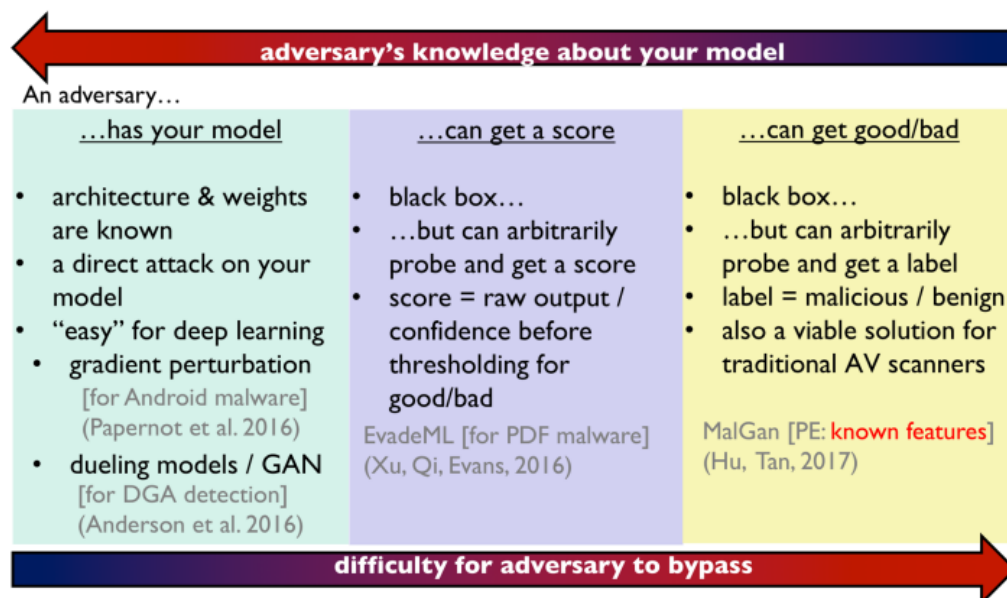


Figure 1: Attack categories, categorized into three coarse bins.

1. Direct gradient-based attacks in which the model must be fully differentiable and the structure and weights must be known by the attacker. Given this, the attacker can essentially query the model directly to determine how best to bypass it.

2. Attacks against models that report a score. The attacker has no knowledge about the model structure, but has unlimited access to probe the model and may be able to learn how to decrease the score.

3. Binary black-box attacks. The attacker has no knowledge about the model, but has unlimited access to probe the model.[5]

### Survey 6:

To avoid these attacks this approach presents further limitations on the information available to an attacker. 1. Output from the target classifier is strictly Boolean, declaring only whether a sample is deemed benign or malicious by the classifier. 2. The feature space and structure of the target classifier are completely unknown. 3. There does not exist an external party (such as an oracle) to guarantee that a sample is valid. Thus, there is no mapping function to the space of legitimate PE files. These restrictions present what we believe is the most difficult black-box evasion scenario from an attacker's perspective. As a result of the limited information available to the attacker, evasion rates are significantly lower than those of the approaches above.[6]

### Algorithms:

In this project we will present the functionality and accuracy of various machine learning algorithms to detect malware. The machine learning algorithms that we will be focusing on are Decision Tree, Support Vector Machine (SVM), Random Forest and XGBoost. The implementation would be carried out in python.

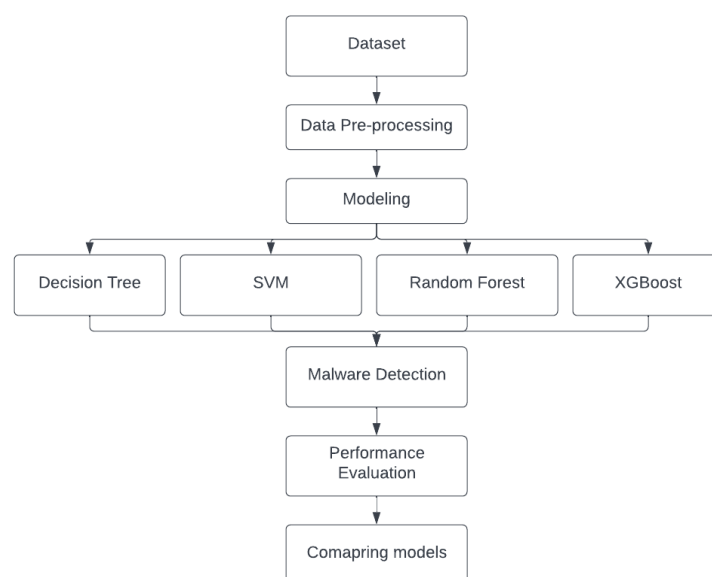


Fig: Our approach / architecture flow diagram

## Code :

### Random Forest:

```
# Random Forest

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# Importing the dataset

dataset = pd.read_csv('data.csv', sep = '|')

X = dataset.drop(['Name', 'md5', 'legitimate'], axis = 1).values

y = dataset['legitimate'].values

# Splitting the dataset into the Training set and Test set

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)

# Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X_train = sc.fit_transform(X_train)

X_test = sc.transform(X_test)

from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(n_estimators = 50, criterion = 'entropy', random_state = 0)

classifier.fit(X_train, y_train)

#predict the test results
```

```
y_pred = classifier.predict(X_test)

#Making the confusion matrix

from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)

#Applying K-Fold cross validation

from sklearn.model_selection import cross_val_score

accuracies = cross_val_score(estimator = classifier, X = X_train, y = y_train, cv = 2)

print(accuracies.mean())

print(accuracies.std())
```

## **Decision Tree:**

```
# Decision Tree

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# Importing the dataset

dataset = pd.read_csv('data.csv', sep = '|')

X = dataset.drop(['Name', 'md5', 'legitimate'], axis = 1).values

y = dataset['legitimate'].values

# Splitting the dataset into the Training set and Test set

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)

# Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X_train = sc.fit_transform(X_train)

X_test = sc.transform(X_test)
```



```

from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier()

clf = clf.fit(X_train,y_train)

#predict the test results

y_pred = clf.predict(X_test)

#Making the confusion matrix

from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)

#Applying K-Fold cross validation

from sklearn.model_selection import cross_val_score

accuracies = cross_val_score(estimator = clf, X = X_train, y = y_train, cv = 2)

print(accuracies.mean())

print(accuracies.std())

```

## **SVM:**

```

# SVM

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# Importing the dataset

dataset = pd.read_csv('data.csv', sep = '|')

X = dataset.drop(['Name', 'md5', 'legitimate'], axis = 1).values

y = dataset['legitimate'].values

# Splitting the dataset into the Training set and Test set

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)

# Feature Scaling

```

```

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X_train = sc.fit_transform(X_train)

X_test = sc.transform(X_test)

from sklearn.svm import SVC

clf = SVC(kernel='linear')

# fitting x samples and y classes

clf.fit(X_train, y_train)

#predict the test results

y_pred = clf.predict(X_test)

#Making the confusion matrix

from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)

cm

#Applying K-Fold cross validation

from sklearn.model_selection import cross_val_score

accuracies = cross_val_score(estimator = clf, X = X_train, y = y_train, cv = 2)

print(accuracies.mean())

print(accuracies.std())

```

## **XGBoost:**

```

# XGboost

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# Importing the dataset

dataset = pd.read_csv('data.csv', sep = '|')

```

```
X = dataset.drop(['Name', 'md5', 'legitimate'], axis = 1).values
y = dataset['legitimate'].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

#Fitting xgboost to the training Set
from xgboost import XGBClassifier
classifier = XGBClassifier(max_depth=20, learning_rate=0.3, n_estimators=150)
classifier.fit(X_train, y_train)

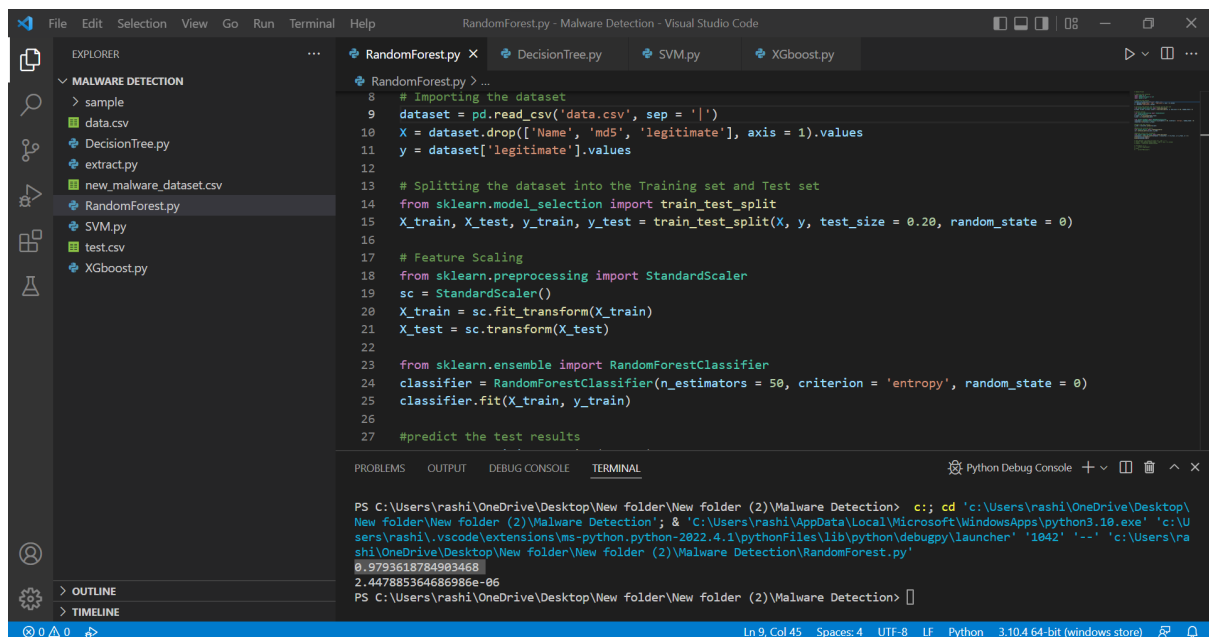
#predict the test results
y_pred = classifier.predict(X_test)

#Making the confusion matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

#Applying K-Fold cross validation
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = X_train, y = y_train, cv = 10)
print(accuracies.mean())
print(accuracies.std())
```

# Screenshots :

## Random Forest:



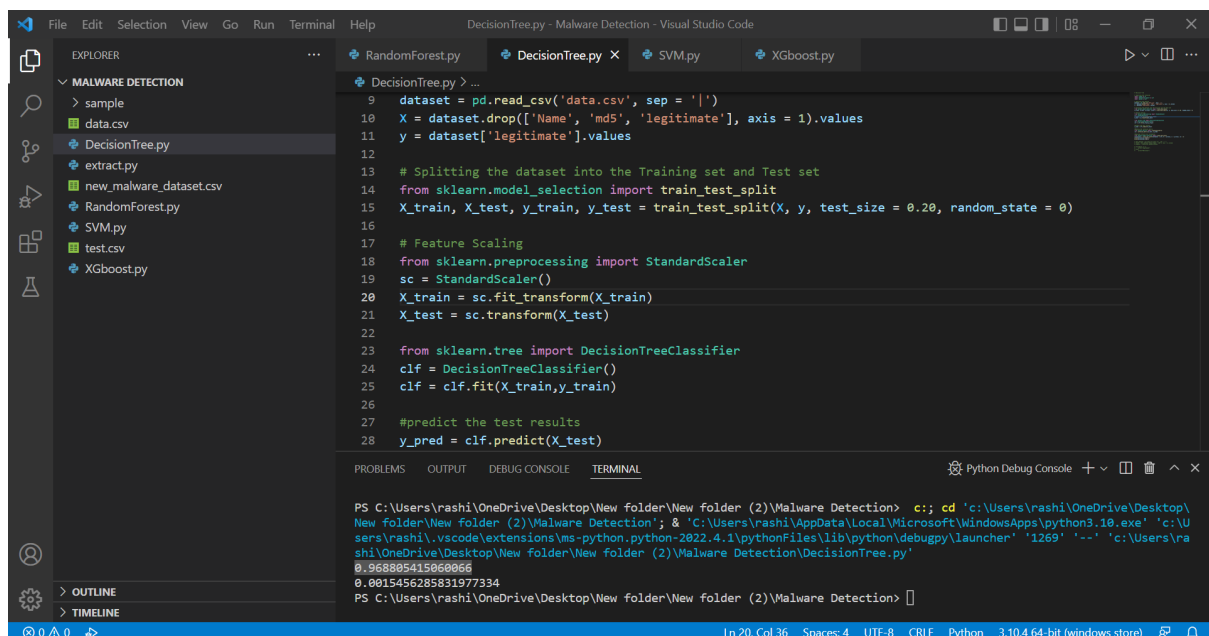
```
File Edit Selection View Go Run Terminal Help
RandomForest.py - Malware Detection - Visual Studio Code

EXPLORER
MALWARE DETECTION
  sample
  data.csv
  DecisionTree.py
  extract.py
  new_malware_dataset.csv
  RandomForest.py
  SVM.py
  test.csv
  XGboost.py

RandomForest.py > ...
8 # Importing the dataset
9 dataset = pd.read_csv('data.csv', sep = '|')
10 X = dataset.drop(['Name', 'md5', 'legitimate'], axis = 1).values
11 y = dataset['legitimate'].values
12
13 # Splitting the dataset into the Training set and Test set
14 from sklearn.model_selection import train_test_split
15 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)
16
17 # Feature Scaling
18 from sklearn.preprocessing import StandardScaler
19 sc = StandardScaler()
20 X_train = sc.fit_transform(X_train)
21 X_test = sc.transform(X_test)
22
23 from sklearn.ensemble import RandomForestClassifier
24 classifier = RandomForestClassifier(n_estimators = 50, criterion = 'entropy', random_state = 0)
25 classifier.fit(X_train, y_train)
26
27 #predict the test results
28
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Python Debug Console
PS C:\Users\rashi\OneDrive\Desktop\New folder\New folder (2)\Malware Detection> c:; cd 'c:\Users\rashi\OneDrive\Desktop\New folder\New folder (2)\Malware Detection'; & 'C:\Users\rashi\AppData\Local\Microsoft\WindowsApps\python3.10.exe' 'c:\Users\rashi\.vscode\extensions\ms-python.python-2022.4.1\pythonFiles\lib\python\debugpy\launcher' '1042' '--' 'c:\Users\rashi\OneDrive\Desktop\New folder\New folder (2)\Malware Detection\RandomForest.py'
0.9793618784903468
2.447885364686986e-06
PS C:\Users\rashi\OneDrive\Desktop\New folder\New folder (2)\Malware Detection>

Ln 9, Col 45 Spaces: 4 UTF-8 LF Python 3.10.4 64-bit (windows store)
```

## Decision Tree:



```
File Edit Selection View Go Run Terminal Help
DecisionTree.py - Malware Detection - Visual Studio Code

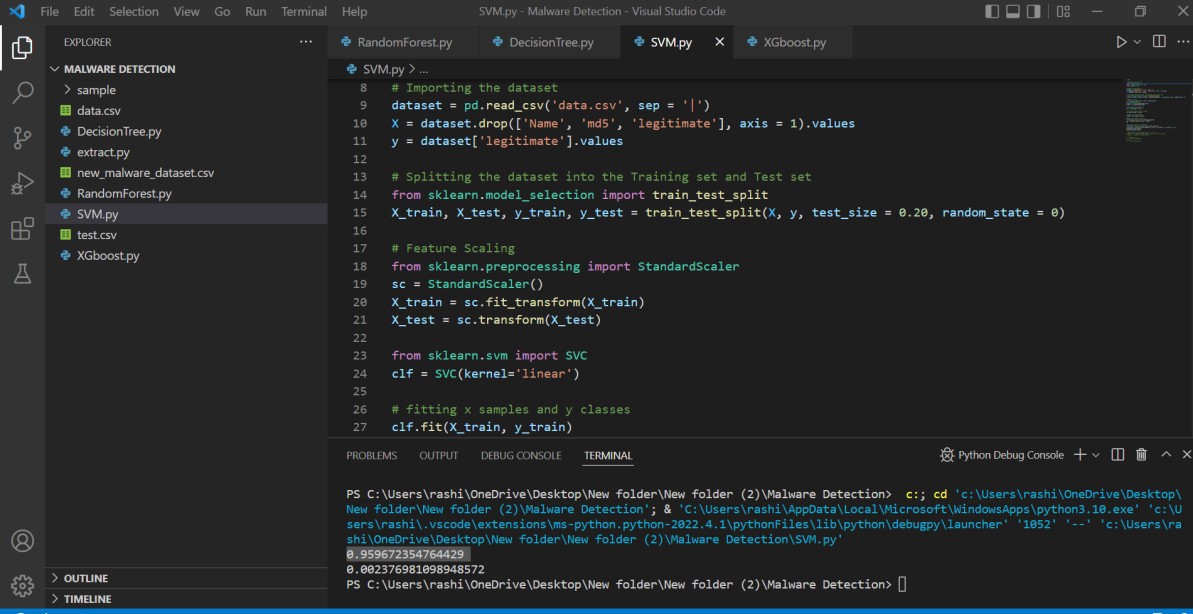
EXPLORER
MALWARE DETECTION
  sample
  data.csv
  DecisionTree.py
  extract.py
  new_malware_dataset.csv
  RandomForest.py
  SVM.py
  test.csv
  XGboost.py

DecisionTree.py > ...
9 dataset = pd.read_csv('data.csv', sep = '|')
10 X = dataset.drop(['Name', 'md5', 'legitimate'], axis = 1).values
11 y = dataset['legitimate'].values
12
13 # Splitting the dataset into the Training set and Test set
14 from sklearn.model_selection import train_test_split
15 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)
16
17 # Feature Scaling
18 from sklearn.preprocessing import StandardScaler
19 sc = StandardScaler()
20 X_train = sc.fit_transform(X_train)
21 X_test = sc.transform(X_test)
22
23 from sklearn.tree import DecisionTreeClassifier
24 clf = DecisionTreeClassifier()
25 clf = clf.fit(X_train, y_train)
26
27 #predict the test results
28 y_pred = clf.predict(X_test)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Python Debug Console
PS C:\Users\rashi\OneDrive\Desktop\New folder\New folder (2)\Malware Detection> c:; cd 'c:\Users\rashi\OneDrive\Desktop\New folder\New folder (2)\Malware Detection'; & 'C:\Users\rashi\AppData\Local\Microsoft\WindowsApps\python3.10.exe' 'c:\Users\rashi\.vscode\extensions\ms-python.python-2022.4.1\pythonFiles\lib\python\debugpy\launcher' '1269' '--' 'c:\Users\rashi\OneDrive\Desktop\New folder\New folder (2)\Malware Detection\DecisionTree.py'
0.968805415060066
0.0015456285831977334
PS C:\Users\rashi\OneDrive\Desktop\New folder\New folder (2)\Malware Detection>

Ln 20, Col 36 Spaces: 4 UTF-8 CRLF Python 3.10.4 64-bit (windows store)
```

## SVM:

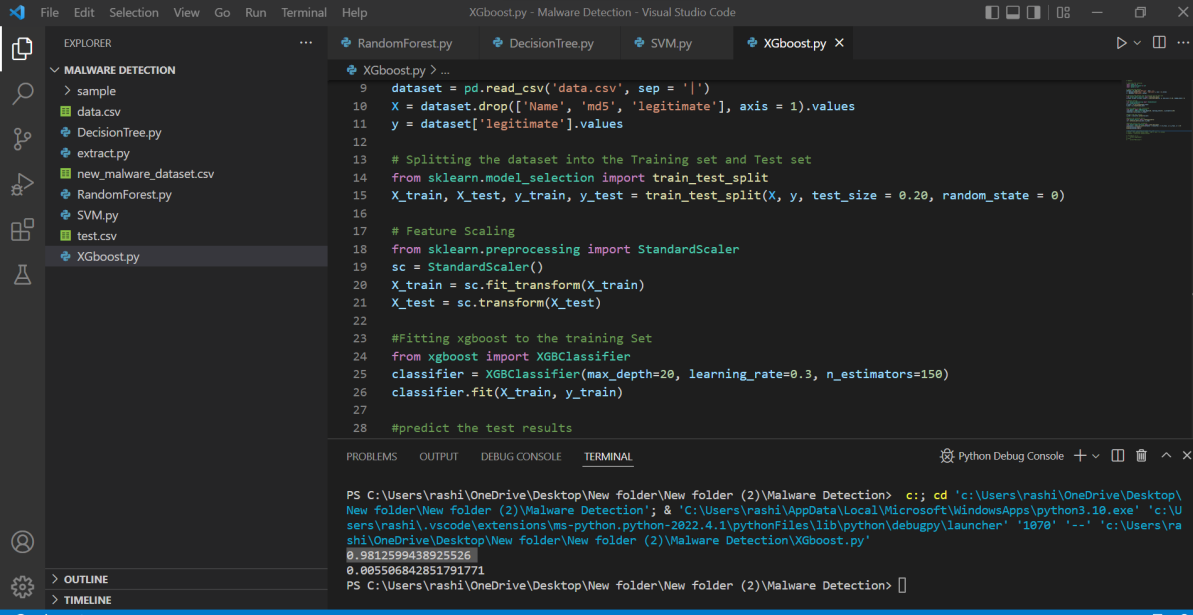


The screenshot shows the Visual Studio Code interface with a project named 'MALWARE DETECTION'. The Explorer panel on the left lists files: sample, data.csv, DecisionTree.py, extract.py, new\_malware\_dataset.csv, RandomForest.py, SVM.py (selected), test.csv, and XGboost.py. The main editor displays the code for SVM.py, which includes importing the dataset, splitting it into training and testing sets, feature scaling, and training an SVM classifier. The terminal at the bottom shows the command to run the script, which executes successfully.

```
SVM.py > ...
8 # Importing the dataset
9 dataset = pd.read_csv('data.csv', sep = '|')
10 X = dataset.drop(['Name', 'md5', 'legitimate'], axis = 1).values
11 y = dataset['legitimate'].values
12
13 # Splitting the dataset into the Training set and Test set
14 from sklearn.model_selection import train_test_split
15 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)
16
17 # Feature Scaling
18 from sklearn.preprocessing import StandardScaler
19 sc = StandardScaler()
20 X_train = sc.fit_transform(X_train)
21 X_test = sc.transform(X_test)
22
23 from sklearn.svm import SVC
24 clf = SVC(kernel='linear')
25
26 # fitting x samples and y classes
27 clf.fit(X_train, y_train)
```

```
PS C:\Users\rashi\OneDrive\Desktop\New folder\New folder (2)\Malware Detection> c:\cd 'c:\Users\rashi\OneDrive\Desktop\New folder\New folder (2)\Malware Detection'; & 'C:\Users\rashi\AppData\Local\Microsoft\WindowsApps\python3.10.exe' 'c:\Users\rashi\.vscode\extensions\ms-python.python-2022.4.1\pythonFiles\lib\python\debugpy\launcher' '1052' '--' 'c:\Users\rashi\OneDrive\Desktop\New folder\New folder (2)\Malware Detection\SVM.py'
0.959672354764429
0.002376981098948572
PS C:\Users\rashi\OneDrive\Desktop\New folder\New folder (2)\Malware Detection>
```

## XGBoost:



The screenshot shows the Visual Studio Code interface with the same 'MALWARE DETECTION' project. The Explorer panel now highlights XGboost.py. The main editor displays the code for XGboost.py, which follows a similar structure to the SVM code but uses XGBoost's XGBClassifier. The terminal shows the command to run the script, which executes successfully.

```
XGboost.py > ...
9 dataset = pd.read_csv('data.csv', sep = '|')
10 X = dataset.drop(['Name', 'md5', 'legitimate'], axis = 1).values
11 y = dataset['legitimate'].values
12
13 # Splitting the dataset into the Training set and Test set
14 from sklearn.model_selection import train_test_split
15 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)
16
17 # Feature Scaling
18 from sklearn.preprocessing import StandardScaler
19 sc = StandardScaler()
20 X_train = sc.fit_transform(X_train)
21 X_test = sc.transform(X_test)
22
23 #Fitting xgboost to the training Set
24 from xgboost import XGBClassifier
25 classifier = XGBClassifier(max_depth=20, learning_rate=0.3, n_estimators=150)
26 classifier.fit(X_train, y_train)
27
28 #predict the test results
```

```
PS C:\Users\rashi\OneDrive\Desktop\New folder\New folder (2)\Malware Detection> c:\cd 'c:\Users\rashi\OneDrive\Desktop\New folder\New folder (2)\Malware Detection'; & 'C:\Users\rashi\AppData\Local\Microsoft\WindowsApps\python3.10.exe' 'c:\Users\rashi\.vscode\extensions\ms-python.python-2022.4.1\pythonFiles\lib\python\debugpy\launcher' '1070' '--' 'c:\Users\rashi\OneDrive\Desktop\New folder\New folder (2)\Malware Detection\XGboost.py'
0.9812599438925526
0.005506842851791771
PS C:\Users\rashi\OneDrive\Desktop\New folder\New folder (2)\Malware Detection>
```

## Results :

This Project includes using different types of Machine Learning algorithms of finding accuracy and prediction of finding malware in information(Datasets).

We have proposed a malware detection module based on advanced data mining and machine learning. While such a method may not be suitable for home users, being very processor heavy, this can be implemented at enterprise gateway level to act as a central antivirus engine to supplement antiviruses present on end user computers. This will not only easily detect known viruses, but act as a knowledge that will detect newer forms of harmful files. While a costly model requires costly infrastructure, it can help in protecting invaluable enterprise data from security threats, and prevent immense financial damage.

We have implemented four models - Random Forest, Decision Tree, SVM and XGBoost. The accuracy for Random Forest was 0.9793 whereas the accuracy for Decision Tree was 0.9678. The accuracy for SVM was 0.9596 whereas the accuracy for XGBoost was 0.9812. XGBoost gave us the best performance.

## Conclusion:

Malware is a critical threat to user computer systems in terms of stealing confidential information or disabling security. This project presents some of the existing machine learning algorithms directly applied on the data or datasets of malware. It explains how the algorithms will play a role in detecting malware with high accuracy and predictions. We are also using data science and data mining techniques to overcome the drawbacks of existing systems. After implementing various models (Random Forest, Decision Tree, SVM and XGBoost) we found out that XGBoost gives the best performance with accuracy of 0.9812. So, developing a malware detection system using XGBoost can help us to detect malware accurately.

## References:

1. A. Sharma and S. K. Sahay, "Evolution and Detection of Polymorphic and Metamorphic Malware: A Survey," *International Journal of Computer Application*, vol. 90, no. 2, pp. 7–11, 2014.

Link: <https://arxiv.org/ftp/arxiv/papers/1903/1903.02966.pdf>

2. Saraiva, D.A.; Leithardt, V.R.Q.; de Paula, D.; Sales Mendes, A.; González, G.V.; Crocker, P. Prismic: Comparison of symmetric key algorithms for iot devices. *Sensors* 2019, 19, 4312. [CrossRef] [PubMed]

Link:

<https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjZ5K72-dz2AhXtTmwGHQ0pD-oQFnoECElQAQ&url=https%3A%2F%2Fwww.mdpi.com%2F2079-9292%2F10%2F14%2F1694%2Fpdf&usg=AOvVaw2WRZnLFVl3ApP-IatWQCCG>

3. Syuhada Selamat, N. and Mohd Ali, F., 2019. Comparison of malware detection techniques using machine learning algorithm. Indonesian Journal of Electrical Engineering and Computer Science, 16(1), p.435. ISSN 2502-4752, DOI: 10.11591/ijeecs.v16.i1.pp435-440.

Link: <http://doi.org/10.11591/ijeecs.v16.i1.pp435-440>

4. Moubarak, J. and Feghali, T., 2020. Comparing Machine Learning Techniques for Malware Detection. In Proceedings of the 6th International Conference on Information Systems Security and Privacy - ForSE, ISBN 978-989-758-399-5; ISSN 2184-4356, pages 844-851. DOI: 10.5220/0009373708440851.

Link: <https://www.scitepress.org/Link.aspx?doi=10.5220/0009373708440851>

5. The rise of machine learning for detection and classification of malware: Research developments, trends and challenges(2020) - Daniel Gibert \*, Carles Mateu, Jordi Planes.

Link: <https://www.sciencedirect.com/science/article/pii/S1084804519303868?via%3Dihub>.

6. Evading Machine Learning Malware Detection - Hyrum S. Anderson Endgame, Inc. , Anant Kharkar University of Virginia., Bobby Filar Endgame, Inc., Phil Roth Endgame, Inc.

Link: <https://www.blackhat.com/docs/us-17/thursday/us-17-Anderson-Bot-Vs-Bot-Evading-Machine-Learning-Malware-Detection-wp.pdf>

7. Hu, T. and Song, T., 2019. Research on XGboost academic forecasting and analysis modelling. Journal of Physics: Conference Series, 1324(1), p.012091. DOI: 10.1088/1742-6596/1324/1/012091.

Link: <https://iopscience.iop.org/article/10.1088/1742-6596/1324/1/012091>