OBJECTIVES:

- Our project named "Malware Detection Using Machine Learning Algorithms" is aimed at detecting the malware infected portable executable files using machine learning algorithms.
- Malware analysis and prevention methods are increasingly becoming necessary for computer systems connected to the Internet.
- This software exploits the system's vulnerabilities to steal valuable information without the user's knowledge, and stealthily send it to remote servers controlled by attackers.
- These malware needs to be detected so that the user gets an information that his system is getting corrupted, and he can therefore take appropriate measures to prevent further attacks of malware and, he can take some steps to recover from the attacks.
- Our Designed Tool should detect malware and should pop up a notification with "Malware" message immediately,
- If there is no malware in the PE file, it should pop up a notification with "Safe" message and If the file is not the form of PE (i.e .exe,.acm,.efi,.dll,.drv,.sys),it should pop up a notification with "Invalid" message.
- Machine learning has been recently introduced into the field of Malware Detection. Many algorithms have been used which results in differing accuracies in predicting weather the input files are malware or not.
- Many different algorithms like Apache Spark and TuriGraph Lab has been used to predict the malware infected files, but the accuracy is less than 90%. Moreover, the algorithms have become non-existent in today's world.

Literature Survey:

- Machine Learning-Based Malware Detection Techniques:

    1. Krugel et al. used dynamic analysis to detect obfuscated malicious code using a mining algorithm.
    2. Authors in proposed a hybrid model for the detection of malware using different features like byte n-gram, assembly n-gram, and library functions to classify an executable as malware or benign.
    3. The work considers the system call subsequence as an element and regards the co-occurrence of system calls as features to describe the dependent relationship between system calls.
    4. Furthermore, the work in extracted 11 types of static features and employed multiple classifiers in a majority vote fusion approach where classifiers such as SVM, k-NN, naive Bayes, Classification and Regression tree (CART), and Random Forest were used. Nataraj et al. consider the Gabor filter and evaluated it on 25 86 malicious families. Thus, they built a model using the k-nearest Neighbors approach with Euclidean distance.

- Malware Detection Using Other Techniques:

1. DroidScope uses a customized Android kernel to reconstruct semantic views to collect detailed application execution traces.
2. An approach aimed at detecting Android malware families was presented in the method is based on the analysis of system calls sequences and is tested obtaining an accuracy of 97% in mobile malware identification using a 3-gram syscall as a feature.
3. Android malware detection exploiting a set of static features was addressed in unsupervised machine learning techniques were used to build models with the considered feature set, statically obtained from permission invocations, strings, and code patterns.
4. Furthermore, the Alde framework employs static analysis and dynamic analysis to detect the actions of users collected by analytics libraries.
5. Moreover, Alde analyses gives insight into what private information can be leaked by apps that use the same analytics library.
6. Casolare et al. also focused on the Android environment by proposing a model checking-based approach for detecting colluding between Android applications.

Author	Approach	Drawback
M. Christodorescu	Mine malicious behaviour present in a known malware.	The impact of test program choices on the quality of mined malware behaviour was not clear.
M. K. Alzaylaee	Deep learning system that detects malicious Android applications through dynamic analysis using stateful input generation.	Investigation on recent intrusion detection systems were not available.
G. Canfora	Android malware detection method based on sequences of system calls.	Assumption that malicious behaviors are implemented by specific system calls
W.Wang	Framework to effectively and efficiently detect malicious apps	Require datasets of features extracted from malware and harmless samples

L. Nataraj	Effective method for visualizing and classifying malware using image processing	Path to a broader spectrum of novel ways to analyze malware was not fully explored.
S. Ni	Classification algorithm that uses static features called Malware Classification	Time required for malware detection and classification was comparatively more
B. Zhang	Detect unknown malicious executables code using fuzzy pattern recognition.	Fuzzy pattern recognition algorithm suffers from a low detection
H. M. Sun	Detecting worms and other malware by using sequences of WinAPI calls.	Approach is limited to the detection of worms and exploits the use of hardcoded addresses
J. Bergeron	Proposed a slicing algorithm for disassembling binary executables.	Graphs created are huge in size, thus the model is not computationally feasible
Q. Zhang	Approach for recognizing metamorphic malware by using fully automated static analysis	Absence of analysis of the parameters passed to library
V. S. Sathyanarayan	Static extraction to extract API calls from known malware	Detection of malware families does not work for packed malware.

EXPERIMENTS CARRIED:

- In this project we are trying to identify the malware infected program files. Usually malware detection is done through anti-virus software which compares the program to known malwares. But here we are trying to detect malwares using Machine Learning Algorithms i.e. by using the known features of malware and training a model to detect malwares. We are training the malware dataset using 5 ML algorithms:

    1. Random Forest
    2. Decision Tree
    3. Gradient Boosting
    4. GaussianNB
    5. Linear Regression

- Depending on which model gives the highest accuracy in detecting malwares, we will build a Tool, using that specific algorithm, which can be used to detect malware infected files.

PROCEDURES:

Step 1: Collecting and Importing Dataset: Dataset described above is imported using Pandas library.

Step 2: Data Pre-processing: The CSV files contain unimportant parameters like Name of the file, md5 of file that needs to be removed before training the model from its features. So, the Independent variable consist of 13 features listed above while Dependent variable consist of binary digit 1/0 (Legitimate or Malicious).

Step 3: Splitting the dataset: We have used the train_test_split function from Scikit Learn Library to split our dataset in the ratio 1:4 as Test set and Train Set respectively.

Step 4: Building the Training Model: Dataset is fit into 5 different machine learning models using Scikit Learn library function. The models under which testing has been done are as follows: -

- Random Forest Classifier: n_estimators = 50
- Decision Tree Classifier: Max_depth = 10
- Gradient Boosting Classifier: n_estimators = 50
- GaussianNB
- Linear Regression

From analysis, 'Random Forest Classifier' gives best accuracy 99.47%. Thus, we will employ Random Forest Classifier as mode of Classification of files as Malicious or Legitimate.

Step 5: Calculation of False Positive and False Negative Rate: Taking Random Forest Model as our classifier we calculate the False Positives and False Negatives of the trained model by making the Confusion Matrix. Apart from Accuracy it is also necessary that for any classifier to be acceptable, false positives and false negatives must be as low as possible.

The results obtained are as follows: -

False positive rate: 0.096150 %

False negative rate: 0.176657 %

Step 6: Final Step – Saving the model: Finally, after training the model, the trained model is saved using JOBLIB and PICKLE. Classifier Model is saved using JOBLIB while Features that are used are saved using PICKLE. Both files are saved in the file extension of ". pkl". This saved model is further used for predicting new results as well as for deploying it in real time platform.

Methodology: Modules-

a. Dataset preparation

b. Training dataset using machine learning algorithms

c. Feature Selection

d. Deploying most accurate algorithm in the application

a. Dataset preparation

1. In order to get started, we first need a set of data on which we can train our algorithms.

2. To create the data set we used file executable greenhouses infested with wax.

3. We downloaded them from **Virrushare** - a repository of malware samples to provide security researchers, incident responders etc. access to sample of live malicious code.

4. The dataset consists of 138048 data entries listing PE Information from Various sources. Out of these 96724 are malicious while 41323 are legitimate. The dataset consists of total **57 features** for every File.

```
[1]: import numpy as np
     import pandas as pd
     import os

[2]: dataset=pd.read_csv('Dataset.csv')

[3]: dataset.head()
```

[3]:

	Name	md5	Machine	SizeOfOptionalHeader	Characteristics	MajorLinkerVersion	MinorLinkerVersion	SizeOfCode	Size
0	memtest.exe	631ea355665f28d4707448e442fbf5b8	332	224	258	9	0	361984	
1	ose.exe	9d10f99a6712e28f8acd5641e3a7ea6b	332	224	3330	9	0	130560	
2	setup.exe	4d92f518527353c0db88a70fddcfd390	332	224	3330	9	0	517120	
3	DW20.EXE	a41e524f8d45f0074fd07805ff0c9b12	332	224	258	9	0	585728	
4	dwtrig20.exe	c87e561258f2f8650cef999bf643a731	332	224	258	9	0	294912	

5 rows × 57 columns

b. Training dataset using machine learning algorithms

Five Machine Learning Algorithms are involved in this project

1. Random Forest:

1. A Random Forest Algorithm is a supervised machine learning algorithm which is extremely popular and is used for Classification and Regression problems in Machine Learning.

2. We know that a forest comprises numerous trees, and the more trees more it will be robust. Similarly, the greater the number of trees in a Random Forest Algorithm, the higher its accuracy and problem-solving ability.

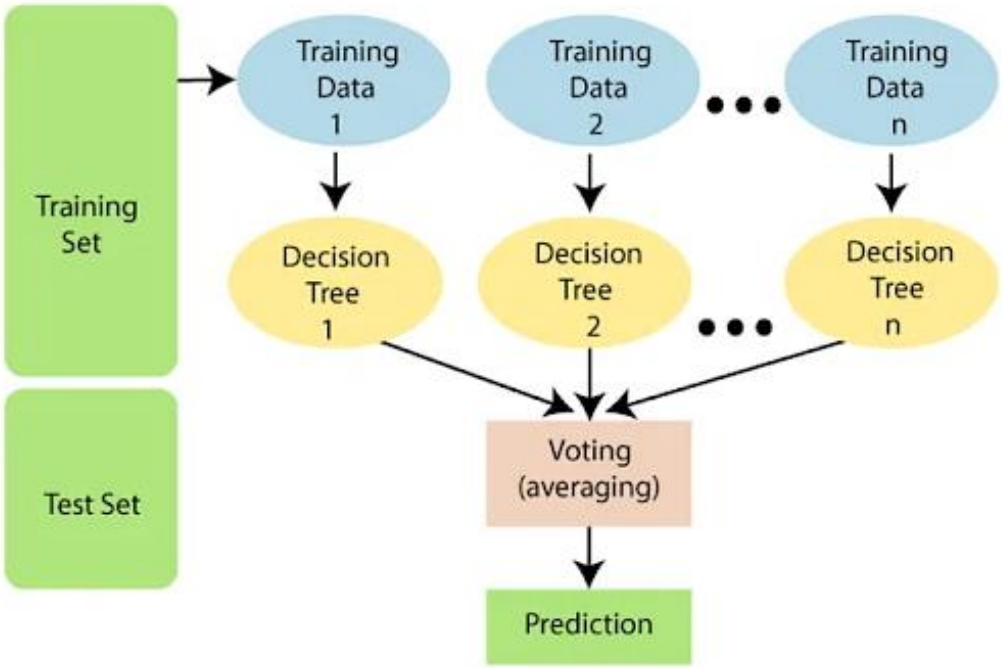
3. Random Forest is a classifier that contains several decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.

4. It is based on the concept of ensemble learning which is a process of combining multiple classifiers to solve a complex problem and improve the performance of the model.

Steps Involved:

- Select random samples from a given data or training
- This algorithm will construct a decision tree for every training data.

- Voting will take place by averaging the decision tree.
- Finally, select the most voted prediction result as the final prediction result.



2. Decision Tree:

1. Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems.

2. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules, and each leaf node represents the outcome.

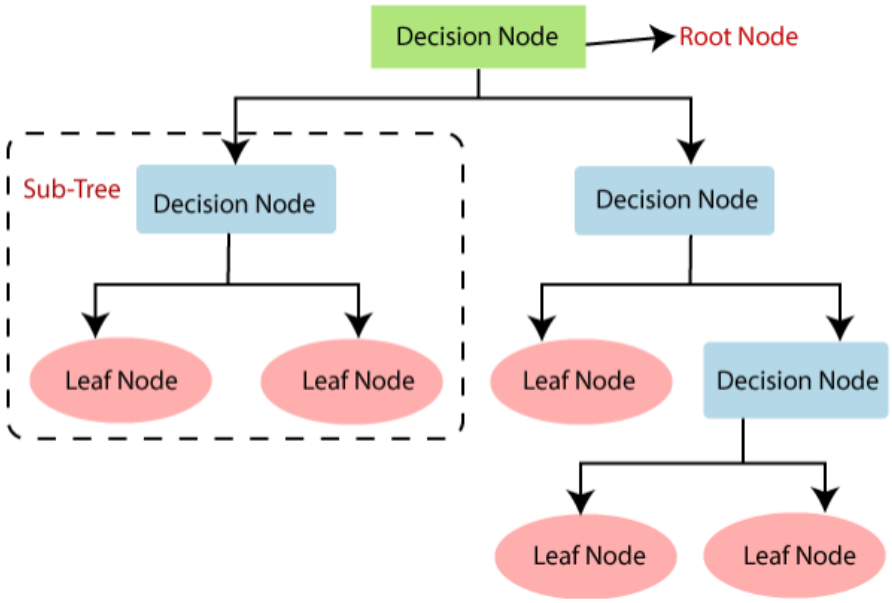
3. In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

4. The decisions or the test are performed based on features of the given dataset.

Steps Involved:

- Begin the tree with the root node, says S, which contains the complete dataset.
- Find the best attribute in the dataset using Attribute Selection Measure
- Divide the S into subsets that contains possible values for the best attributes.

- Generate the decision tree node, which contains the best attribute.
- Recursively make new decision trees using the subsets of the dataset created in step -iii. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.



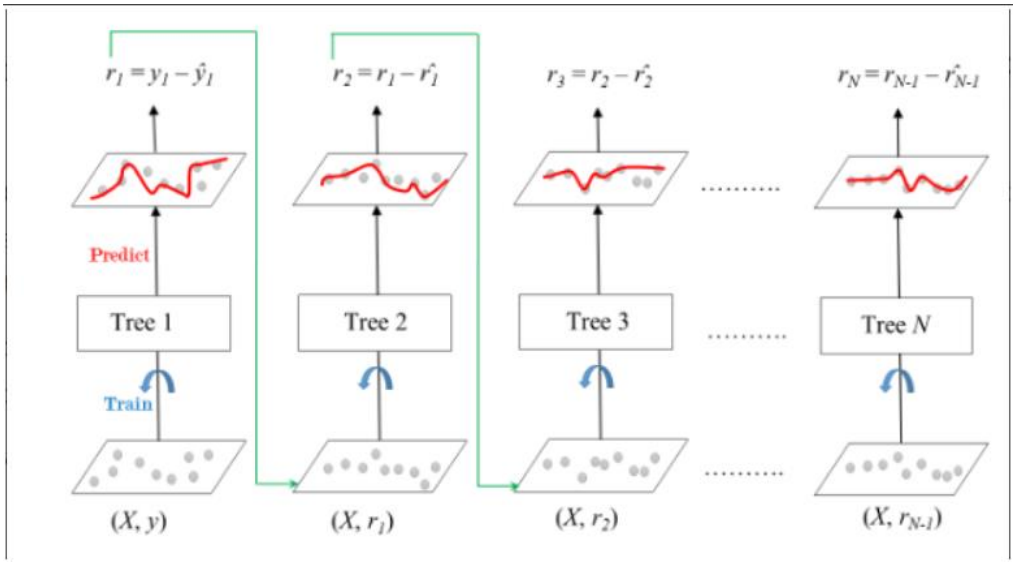
3. Gradient Boosting:

1. Gradient Boosting is a popular boosting algorithm. In gradient boosting, each predictor corrects its predecessor's error.

2. In contrast to Adaboost, the weights of the training instances are not tweaked, instead, each predictor is trained using the residual errors of predecessor as labels.

3. There is a technique called the Gradient Boosted Trees whose base learner is CART (Classification and Regression Trees).

4. The diagram explains how gradient boosted trees are trained for regression problems.



4. Gaussian NB

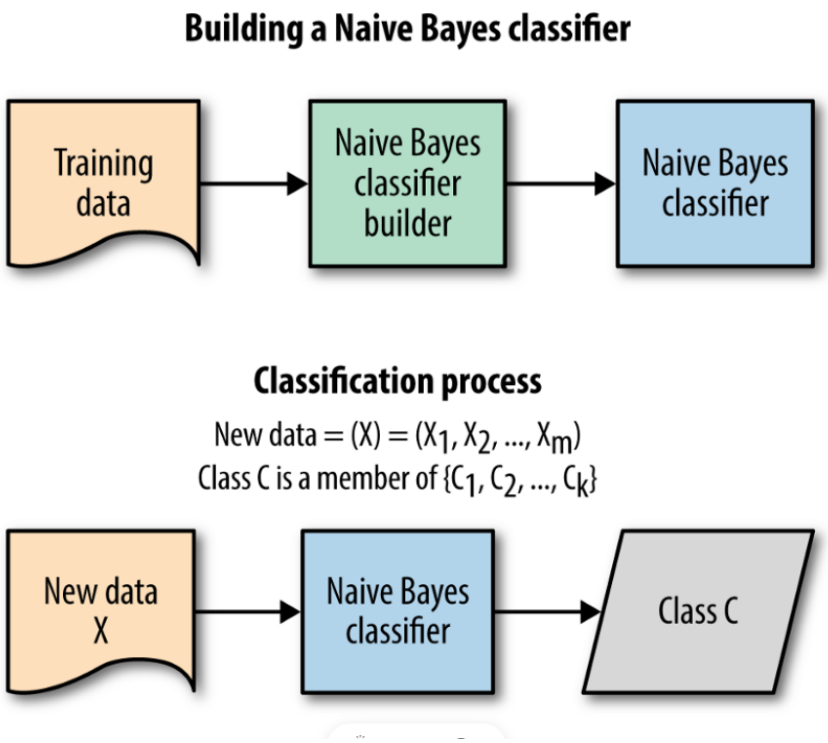
1. **N**aïve Bayes classifier is one of the most effective machine learning algorithms implemented in machine learning projects and distributed MapReduce implementations leveraging Apache Spark.

2. Primarily Naïve Bayes is a linear classifier, which is a supervised machine learning method and works as a probabilistic classifier as well.

3. Most of the time, for the numeric implementations K-Nearest Neighbors and K-Means clustering algorithms can be implemented.

4. Naïve Bayes classifier works effectively for classifying emails, texts, symbols, and names.

5. It's not unusual Naïve Bayes classifier is used for numeric data as well in some instances. Naïve Bayes classifier can be implemented on high-dimensional datasets effectively as well.



5. Linear Regression

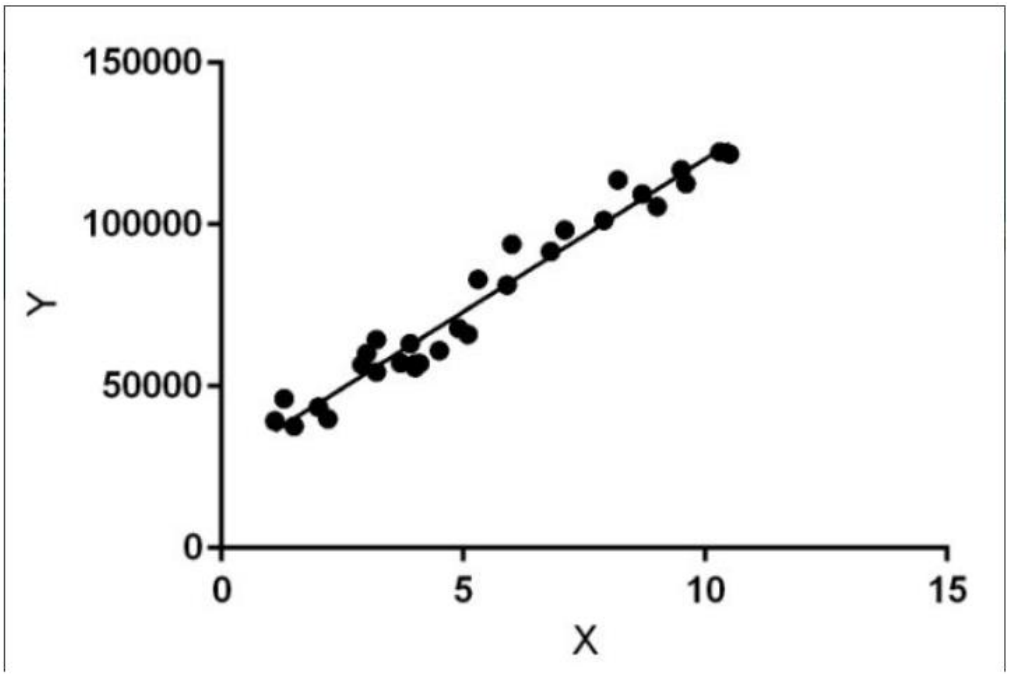
1. Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task.

2. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting.

3. Different regression models differ based on – the kind of relationship between dependent and independent variables they are considering, and the number of independent variables getting used.

Steps Involved:

- Data pre-processing step.
- Fitting Logistic Regression to the Training set.
- Predicting the test result.
- Test accuracy of the result (Creation of Confusion matrix).



c. Feature Selection

1. Taking all features into consideration may lead to overfitting and may produce poor results.

2. In order to find which feature, contribute significantly towards building the model, we employed feature selection algorithm using Extra Tree Classifier.

3. On executing the algorithm, we found that out of 57 features only 13 features were important. Hence only 13 these features were taken into consideration while building the model.

```
1.) feature DllCharacteristics (0.136375)
2.) feature Characteristics (0.115415)
3.) feature Machine (0.104625)
4.) feature VersionInformationSize (0.076241)
5.) feature Subsystem (0.067901)
6.) feature ImageBase (0.053903)
7.) feature SectionsMaxEntropy (0.043873)
8.) feature ResourcesMaxEntropy (0.042044)
9.) feature MajorSubsystemVersion (0.040783)
10.) feature SizeOfOptionalHeader (0.038590)
11.) feature ResourcesMinEntropy (0.038238)
12.) feature MajorOperatingSystemVersion (0.025351)
13.) feature SectionsMinEntropy (0.021914)
```

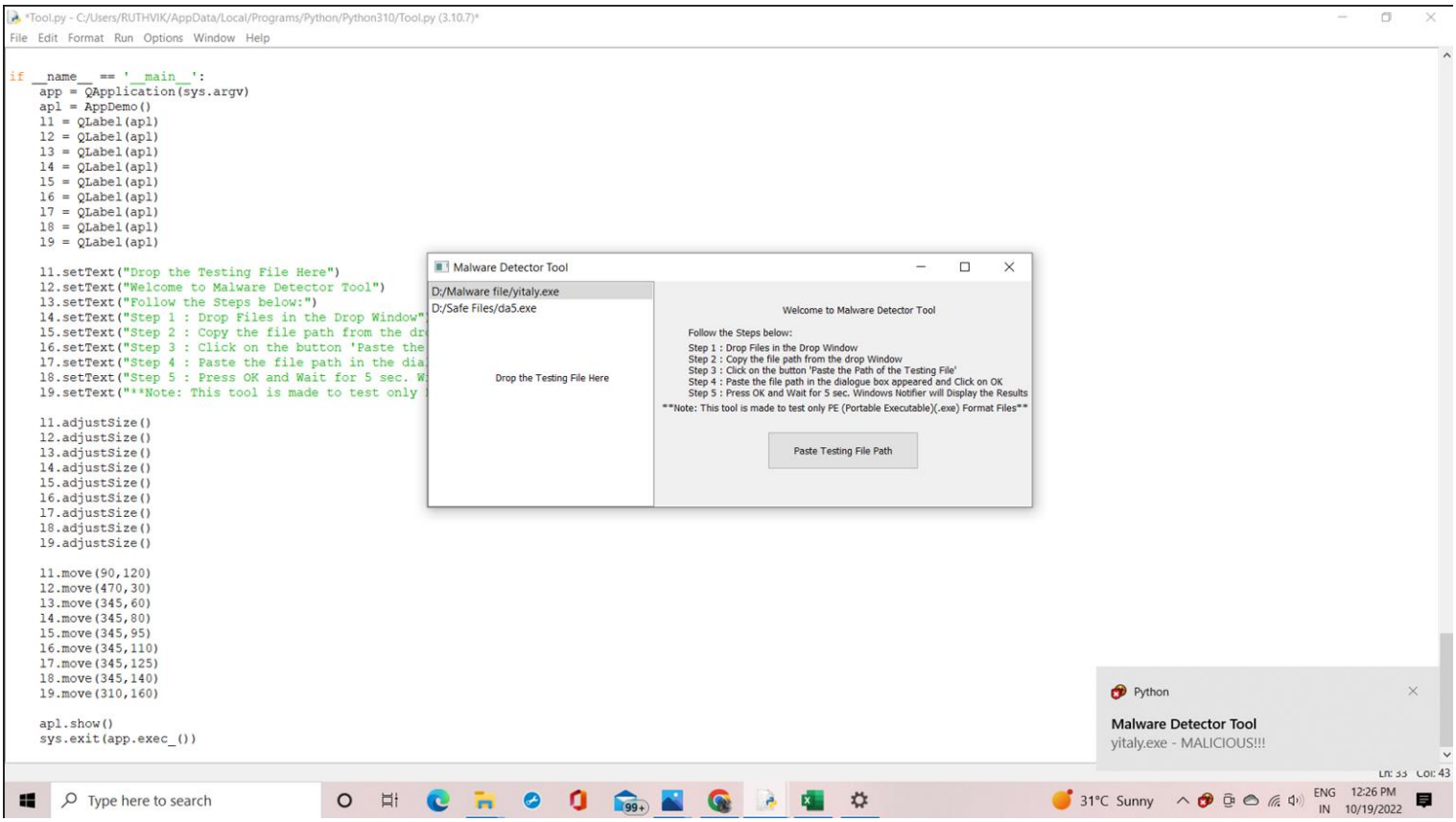
d. Deployment of Application

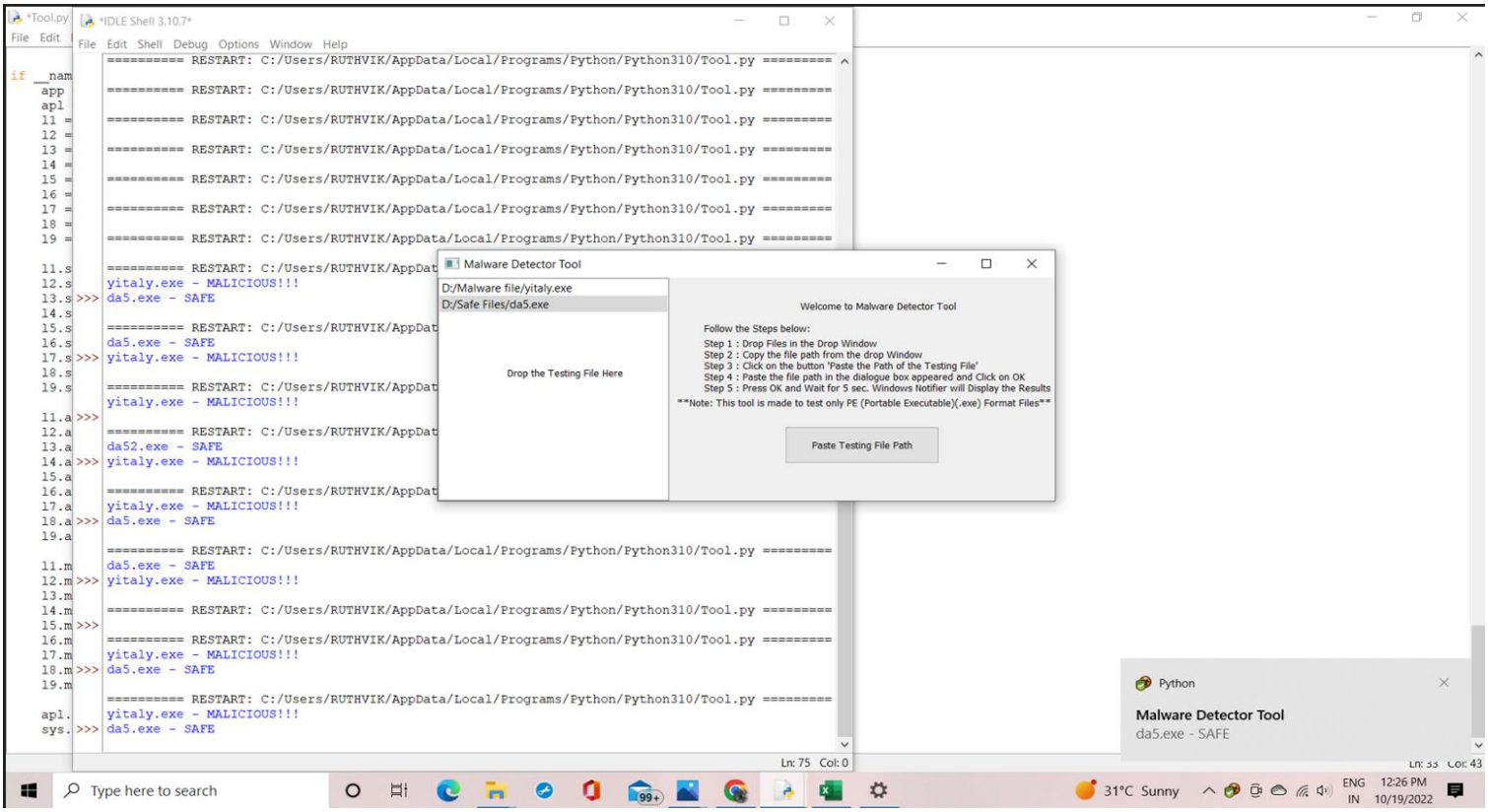
After comparing the precisions of machine learning algorithms, we'll choose the algorithm with the highest accuracy and use it for building the malware detection Tool, through Python-10.7.For Building the Tool the important libraries are Scikit Learn,Numpy,Pandas,Pickle,Pefile,OS,ArgParse,win10toast.

RESULTS OBTAINED:

Method	Accuracy
Random Forest	99.44%
Decision Tree	99.04%
Gradient Boosting	98.80%
GaussianNB	70.09%
Linear Regression	52.47%

```
== RESTART: C:\Users\RUTHVIK\AppData\Local\Programs\Python\Python310\Train.py
1.) feature DllCharacteristics (0.136375)
2.) feature Characteristics (0.115415)
3.) feature Machine (0.104625)
4.) feature VersionInformationSize (0.076241)
5.) feature Subsystem (0.067901)
6.) feature ImageBase (0.053903)
7.) feature SectionsMaxEntropy (0.043873)
8.) feature ResourcesMaxEntropy (0.042044)
9.) feature MajorSubsystemVersion (0.040783)
10.) feature SizeOfOptionalHeader (0.038590)
11.) feature ResourcesMinEntropy (0.038238)
12.) feature MajorOperatingSystemVersion (0.025351)
13.) feature SectionsMinEntropy (0.021914)
RandomForest : 0.9944223107569721
DecisionTree : 0.9904744657732706
GradientBoosting : 0.9880115900036219
GNB : 0.7009779065555958
LinearRegression : 0.5247709728128693
False positive rate : 0.096150 %
False negative rate : 0.176657 %
```


Tool.py - C:/Users/RUTHVIK/AppData/Local/Programs/Python/Python310/Tool.py (3.10.7)
File Edit Format Run Options Window Help
if __name__ == '__main__':
app = QApplication(sys.argv)
apl = AppDemo()
l1 = QLabel(apl)
l2 = QLabel(apl)
l3 = QLabel(apl)
l4 = QLabel(apl)
l5 = QLabel(apl)
l6 = QLabel(apl)
l7 = QLabel(apl)
l8 = QLabel(apl)
l9 = QLabel(apl)
l1.setText("Drop the Testing File Here")
l2.setText("Welcome to Malware Detector Tool")
l3.setText("Follow the Steps below:")
l4.setText("Step 1 : Drop Files in the Drop Window")
l5.setText("Step 2 : Copy the file path from the dr
l6.setText("Step 3 : Click on the button 'Paste the
l7.setText("Step 4 : Paste the file path in the dia
l8.setText("Step 5 : Press OK and Wait for 5 sec. W
l9.setText("**Note: This tool is made to test only
l1.adjustSize()
l2.adjustSize()
l3.adjustSize()
l4.adjustSize()
l5.adjustSize()
l6.adjustSize()
l7.adjustSize()
l8.adjustSize()
l9.adjustSize()
l1.move(90,120)
l2.move(470,30)
l3.move(345,60)
l4.move(345,80)
l5.move(345,95)
l6.move(345,110)
l7.move(345,125)
l8.move(345,140)
l9.move(310,160)
apl.show()
sys.exit(app.exec_())
Malware Detector Tool
D:/Malware file/yitaly.exe
D:/Safe Files/da5.exe
Drop the Testing File Here
Welcome to Malware Detector Tool
Follow the Steps below:
Step 1 : Drop Files in the Drop Window
Step 2 : Copy the file path from the drop Window
Step 3 : Click on the button 'Paste the Path of the Testing File'
Step 4 : Paste the file path in the dialogue box appeared and Click on OK
Step 5 : Press OK and Wait for 5 sec. Windows Notifier will Display the Results
Note: This tool is made to test only PE (Portable Executable)(.exe) Format Files
Paste Testing File Path
Python
Malware Detector Tool
yitaly.exe - MALICIOUS!!!
Ln: 33 Col: 43
Type here to search
31°C Sunny
ENG IN
12:26 PM 10/19/2022


*Tool.py
File Edit
IDLE Shell 3.10.7
File Edit Shell Debug Options Window Help
========== RESTART: C:/Users/RUTHVIK/AppData/Local/Programs/Python/Python310/Tool.py ==========
yitaly.exe - MALICIOUS!!!
>>> da5.exe - SAFE
da5.exe - SAFE
>>> yitaly.exe - MALICIOUS!!!
da52.exe - SAFE
Ln: 75 Col: 0
Malware Detector Tool
D:/Malware file/yitaly.exe
D:/Safe Files/da5.exe
Drop the Testing File Here
Welcome to Malware Detector Tool
Follow the Steps below:
Step 1 : Drop Files in the Drop Window
Step 2 : Copy the file path from the drop Window
Step 3 : Click on the button 'Paste the Path of the Testing File'
Step 4 : Paste the file path in the dialogue box appeared and Click on OK
Step 5 : Press OK and Wait for 5 sec. Windows Notifier will Display the Results
Note: This tool is made to test only PE (Portable Executable)(.exe) Format Files
Paste Testing File Path
Python
Malware Detector Tool
da5.exe - SAFE
Ln: 33 Col: 43
Type here to search
31°C Sunny
ENG IN
12:26 PM 10/19/2022