Detection of Zero-day Malware Using Hybrid Supervised and Un-supervised Machine Learning Algorithms

MSc Research Project
Cyber Security

Mohammed Mustafa Raza Choudhry
Student ID: X20119607

School of Computing
National College of Ireland

Supervisor: Dr. Imran Khan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Mohammed Mustafa Raza Choudhry		
Student ID:	X20119607		
Program:	Msc. Cybersecurity	**Year:**	January 2021
Module:	INTERSHIP		
Supervisor:	Dr. Imran Khan		
Submission Due Date:	31st January 2022 (Final Submission)		
Project Title:	Detection of Zero-day Malware Using Hybrid Supervised and Un-supervised Machine Learning Algorithms		
Word Count: 5154	**Page Count**: 20		

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
<u>ALL</u> internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Mohammed Mustafa Raza Choudhry

Date: 28th January 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Detection of Zero-day Malware Using Hybrid Supervised and Un-supervised Machine Learning Algorithms

Mohammed Mustafa Raza Choudhry
X20119607

Abstract

Malware is one of the main dangers for the present figuring world from the number of various sites circulating malicious application is expanding at very fast rate. Malware examination and anticipation strategies can be progressively used to becoming vital for PC frameworks associated with the Internet. This product tries to gain advantage of the framework's weaknesses and security breach to take significant data without the client's information, and covertly send it to distant servers constrained by assailants. Generally, antimalware items used for marking and identifying known malware. In any case, the mark-based strategy doesn't scale well in distinguishing muddled and stuffed malware. Taking into account that the reason for a issue is regularly best perceived by the concentrating on the primary parts of a code likewise the mental aides, guidance opcode as well as API Call, and so forth In this paper, we have researched the pertinence of the highlights of unloaded vindictive with harmless executables like memory helpers, guidance opcodes, and API to distinguish a component that characterizes the executable. Trials were directed on two datasets utilizing AI and profound learning approaches like Random Forest (RF), KNN, Logistic Regression, Naïve Bayes, Decision Tree. The learning strategies and showed a tweaked profound neural organization that brought about a accuracy of 99.72% and 99.37% on UCI Malware and Virus Share Dataset, individually.

1 Introduction

Malware which is a combination of two words Malicious & Software, is one kind of most ever-growing security danger as well as the identification and classification of malicious files remains a significant space of exploration. In the starting phase in identification is investigation. This includes both either dynamic or static investigation of known malicious application and is normally can be used disconnected with master human info. Outcomes of analysis are refined into a “signature". Other strategy for malware location is the utilization of some static marks to inspect programs after they are stacked and just before execution of software [1], [2]. Lamentably, this can be crushed by the malware which utilizes jumbling. Due to this, dynamic conduct-based location has been proposed [3]. These techniques screen the conduct of the framework by utilizing working with framework or hypervisor-based instrumentation all together to make difference malevolent conduct. Dynamic and static marks can be inferred utilizing either deterministic or measurable methods. Measurable methods dependent on AI like ML and DL Methods are utilized to observe designs comparing to malevolent conduct. Interestingly, deterministic marks are regularly developed through human master investigation [4], [5].

In this Digital Era malware have stroked a huge number of computational devices. Computers and Smart Devices can be compromised, some private and confidential information can be stolen, private networks can be penetrated some critical infrastructures can be stultified with using malicious software or ransomwares. As per Computer Economics due to various malware attacks financial loss has arisen to 13.3 billion in 2006 from 3.3 billion in 1997 [6]. Estimation by Cybersecurity Ventures [7] total loss because of malware attack was three trillion in 2015 and it is expected to cross six trillion by 2021.

Norton, McAfee, Kaspersky, AVG, etc. like antivirus software are mainly used for defeating malware attacks. Generally, this antivirus software is using signature-based methods for detecting malware. A short sequence of bytes which is called signature is used to detect malware. Zeus the toolkit for malware generation is used to regenerate millions of different variants of the same malware by using various techniques.

Research analysts and communities for anti-malware have described various DL and ML based methods for malware detection as well as analysis designing. Two fundamental steps Feature Extraction and Feature reduction have been mainly divided in literature survey. Dynamic analysis and static analysis methods are used for generating features for malware analysis.

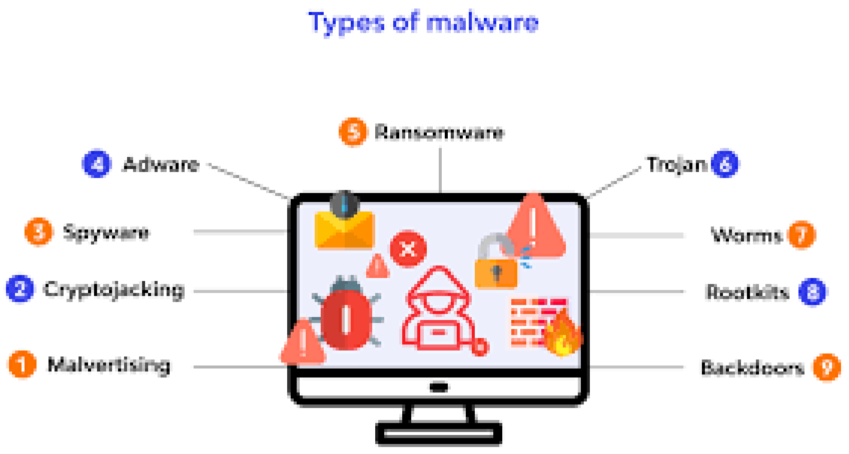


Figure 1: Types of Malwares.

In above figure we can see different types of malwares like Spyware, Ransomware, Trojan, Worms, Adware etc. These can be a jumbling question to define exact type of malware.

Malvertising: In this type attacker uses advertising for spreading the malware. In legitimate advertising network or webpages malicious or malware has been injected.

Spyware: In this type user’s confidential and private information and data is captured and sent to attacker without consent or knowledge of user. Different heterogenous types of spyware are used to track and record victim’s internet usage and also change root level settings in user’s device.

Backdoor: In this method authentication or encryption can be bypassed using independent code or program. Attacker can gain access of user’s confidential and private files using some previously installed or manufacturer program.

Cryptojacking: This is most recent types of malwares in which user’s computing resources and power are used to mine various cryptocurrency without their knowledge.

Adware: This type of malwares is generated to display different unusual ads in user’s device. This is most profitable and least harmful types of malwares.

Ransomware: Your data and computer access can be denied using this type of malware until some ransom is paid. User’s data will be encrypted and converted to not readable form.

2 Related Work

Numerous endeavours have effectively been used to identify malware. In various research papers a few techniques have been used [8]. There are various sorts of malicious files and application recognition and grouping utilizing methods, for example, static, dynamic and half-breed highlights [9]. Static investigation likewise known as code examination [10] without executing malicious application by looking at and noticed for programming code for acquiring data of how malicious application capacities are working. In this something else altogether without utilizing the codes however as indicated by the runtime conduct by watching its framework collaboration, conduct and consequences for have framework called as unique examination [11]. Though half breed examination [12] is a mix of dynamic and static investigation.

By utilizing API calls in his work, Tian et al. [13] have proposed a double component technique for malicious application identification and characterization. In this research article, the authors likewise researched the recurrence put together strategies with respect to similar information however no overhauling was seen over the twofold portrayal. In comparative methodology, Salehi [14] proposed a discovery of malicious files which is dependent on API calls and their contentions. The creators utilized this method as a component and broke down it’s result for the grouping system. To diminish the quantity of elements, include choice calculations are utilized. The outcome from the exploratory assessment defines the exactness of 98.4% by utilizing arbitrary woodland calculation.

M. Singh [15] porjected a new strategy for dependent on social examination on AI that zeroed in on grouping and bunching of malwares. In their research test, they utilized a novel two sorts of classifiers which are Logic Model tree and K-Means calculations. The outcome has displayed that 82% expected for ruining in the PC framework or organization assets in while the rest 18% of investigated malicious files were implanted with a systems administration ability to associate the external world. A. Salim and P.V.Shijo [16] used a strategy which gives the proficient robotized grouping of malwares by utilizing both elements of malwares by utilizing AI procedure. In their analytical trial, the static highlights are extricated from the parallel code while in most powerful investigation is finished by utilizing the device sandbox of cuckoo that zeroed in on framework sequences of calls. The creators designed a new examination with utilizing various static, dynamic and incorporated strategy with utilizing two different classifiers which are SVM and arbitrary forest (RF). The precision discovery shows for coordinated technique in RF 97.68% while 98.71% utilizing SVM calculation.

M.Aizaini [17] projected a novel improvement of decision tree-based calculation for ordering of malicious and harmless. On double class researchers have accomplished exactness 94.66% by utilizing an API in component extraction. Liu [18], utilized Neural organization for malware recognition. The creators generally utilize the static highlights acquired by against aggregation of APK. These static highlights have incorporated string, touchy API, declarations and application authorizations. Yang et al [19], proposed a novel high level arbitrary woodland calculation for identifying and analysing malware. Santos [20] essentially utilized the static element of PE records. In above research paper, author have prescribed another hybrid technique to recognize obscure malware families dependent on the recurrence for the presence of opcode successions.

C. I. Fun [20], proposed strategies for snaring for following dynamic marks that the malicious attempts to stow away by utilizing information mining techniques. This is the procedure which can be used to distinguished various practices of malware and they contrast it and the harmless information. By utilizing 80 credits, we found the discovery rate was 95% which makes the procedure of utilized expanded location rate with the diminishing intricacy. M. Belaoued [21] proposed a constant PE malware location framework dependent on the examination of the data to put away in the PE-discretionary header fields. For highlights choice, the essayists utilized Phi coefficient and chi square with chosen highlights Rotation woods classifier was prepared and tried.

Below table is provided for drawback and used approaches in Table 1.

Table 1: Existing Technique Comparison

Author	Used Approach	Drawback
M. Christodorescu [23]	Mining the unknown and malicious behaviour which is present in well-known malware.	The quality of mined malware behaviour was not known due to impact of test program choices
W.Wang [26]	A novel framework defined for detecting malicious and benign applications.	For training their models they require feature extracted dataset.
M. K. Alzaylaee [24]	Stateful input generation based dynamic analysis is used to detect android application which are malicious.	Recent intrusion detection system investigation was not used.
L. Nataraj [27]	Image processing techniques has been used for classification and visualization of malware.	This border spectrum path is not fully explored
G. Canfora [25]	Various sequence of system calls is used for detecting malware in android application.	Only based on assumptions that specific system calls used for implementing malware behaviour.

3 Research Methodology

Various machine learning algorithms have been used in this research. Mainly there are two types of Supervised and Unsupervised techniques have been used for detection and classification of malicious application.

In this research we have used below five different techniques of machine learning on two different datasets.

3.1 Logistic Regression

This method is used for binary classification either 0 or 1. Like linear regression the goal is to finding the values for coefficients as each input variable have weight and unlike linear regression the output is being transformed using logistic function for prediction of the output.

3.2 Decision Trees

A Decision tree expands upon iteratively posing inquiries to segment information. It is a simpler to conceptualize the dividing information with a visual portrayal of a choice tree. This addresses a choice tree to anticipate client agitate. First split depends on month-to-month charges sum. Then, at that point, the calculation continues to pose inquiries to isolate class marks. The inquiries get more explicit as the tree gets further.

The point of the choice tree calculation is to build the prescience however much as could be expected at each parcelling so the model continues to acquire data about the dataset

3.3 Random Forest

Random Forest is a gathering of numerous choice trees. Random Forest are constructed utilizing a strategy called sacking in which choice trees are utilized as equal assessors. Whenever utilized for a grouping issue, the outcome depends on greater part vote of the outcomes got from every choice tree. For relapse, the expectation of a leaf hub is the mean worth of the objective qualities in that leaf. Arbitrary woods relapse takes mean worth of the outcomes from choice trees.

3.4 Naive Bayes

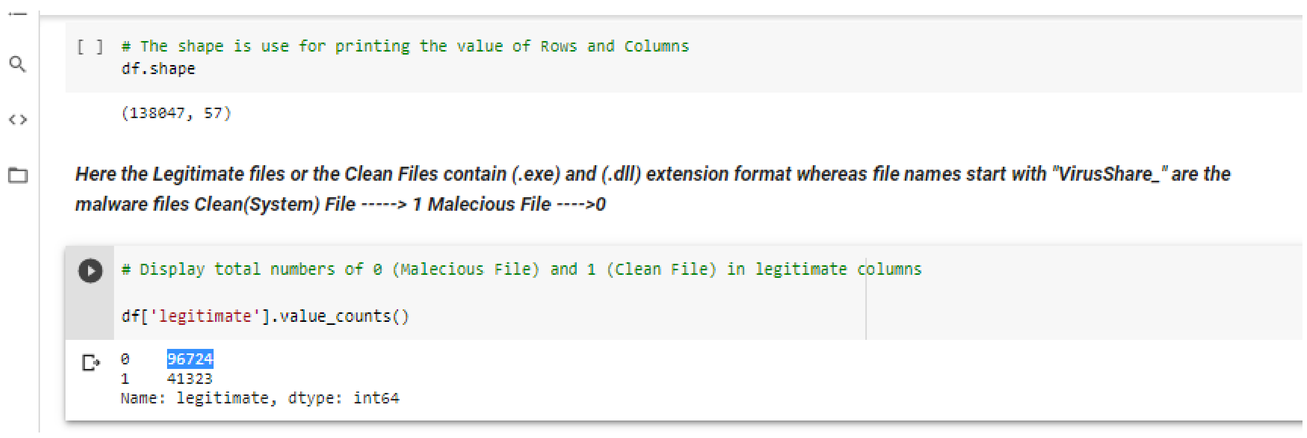
This is a type of regulated learning calculation utilized for characterization errands. Henceforth, it is additionally called Naive Bayes Classifier. Naive bayes expects that highlights are autonomous of one another and there is no connection be tween's elements. Be that as it may, this isn't true, all things considered. This innocent presumption of elements being uncorrelated is the justification for why this calculation is classified "Naive".

3.5 K-Nearest Neighbors (kNN)

K-closest neighbors (kNN) is a type of managed learning calculation that can be mainly utilized to address both arrangement and relapse assignments. The principle can be thought as behind kNN is that the worth or class of an information point is controlled by the information focuses around it. kNN classifier decides the class of an information point by larger part casting a ballot guideline.

A significant variable to the accomplishment of this task is the curation of a dataset that intently takes after the sort of harmless and malevolent doubles as of now available for use. Having such a dataset is fundamental in building AI models that can effectively sum up. We get an underlying dataset of harmless examples from the pairs found on our Windows PCs. These contain some outsider programming yet are overwhelmingly Windows 10 framework les. These are enhanced with doubles taken from Windows Server 2000 and 2012, Windows XP, and Windows 7 establishments. To change up the harmless examples, an extra arrangement of doubles is gotten by introducing the 300 most famous bundles from a Windows bundle director. In the wake of joining the harmless examples and eliminating copies utilizing the Linux utility fdupes for checking MD5 hashes, we are left with 41,323 special pairs.

Vindictive examples are sourced from the malware store VirusShare [28]. Tests transferred to the website inside the most recent two years are downloaded in mass. The Windows pairs are then, at that point, idented and questioned for their Virus Total report which gives the consequence of outputs by 60/70 of the main enemy of infection (AV) programming. We apply a limit of 30 sweeps distinguishing the twofold as vindictive to come to a set of 96 724 special malware tests. So total samples including normal and malicious software 1 38 047 were used for training and testing.



```
[ ] # The shape is use for printing the value of Rows and Columns
    df.shape

    (138047, 57)
```

Here the Legitimate files or the Clean Files contain (.exe) and (.dll) extension format whereas file names start with "VirusShare_" are the malware files Clean(System) File -----> 1 Malecious File ---->0

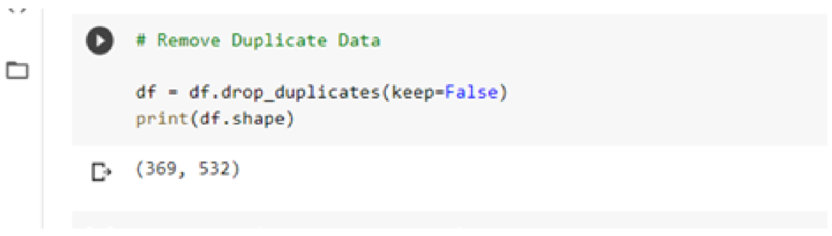
```
# Display total numbers of 0 (Malecious File) and 1 (Clean File) in legitimate columns

df['legitimate'].value_counts()

0    96724
1    41323
Name: legitimate, dtype: int64
```

Figure 2: Virus share Dataset.

We have also used our techniques in another dataset of UCI which was taken from Kaggel. In that dataset for different features, we have used factorization methods and total 369 samples of malicious and benign software were used.



```
# Remove Duplicate Data

df = df.drop_duplicates(keep=False)
print(df.shape)

(369, 532)
```

Figure 3: UCI Dataset.

4 Design Specification

The proposed approach for detection of malware is shown in the below flowchart in Figure 4. A malware test was examined utilizing static examination. Static investigation prompts the features of elements. In this analysis, we inspected malicious and normal file records. This Malicious executable records data then, at that point, will be utilized as an element. All the data elements will then, at that point, be separated to choose ideal elements that are applicable for grouping task. The last cycle will be finished by assessing the most elevated exactness identification utilizing three kinds of calculations which are Logistic regression, Random Forest, K-Nearest Neighbour (KNN), Naïve byes and Decision tree (DT).

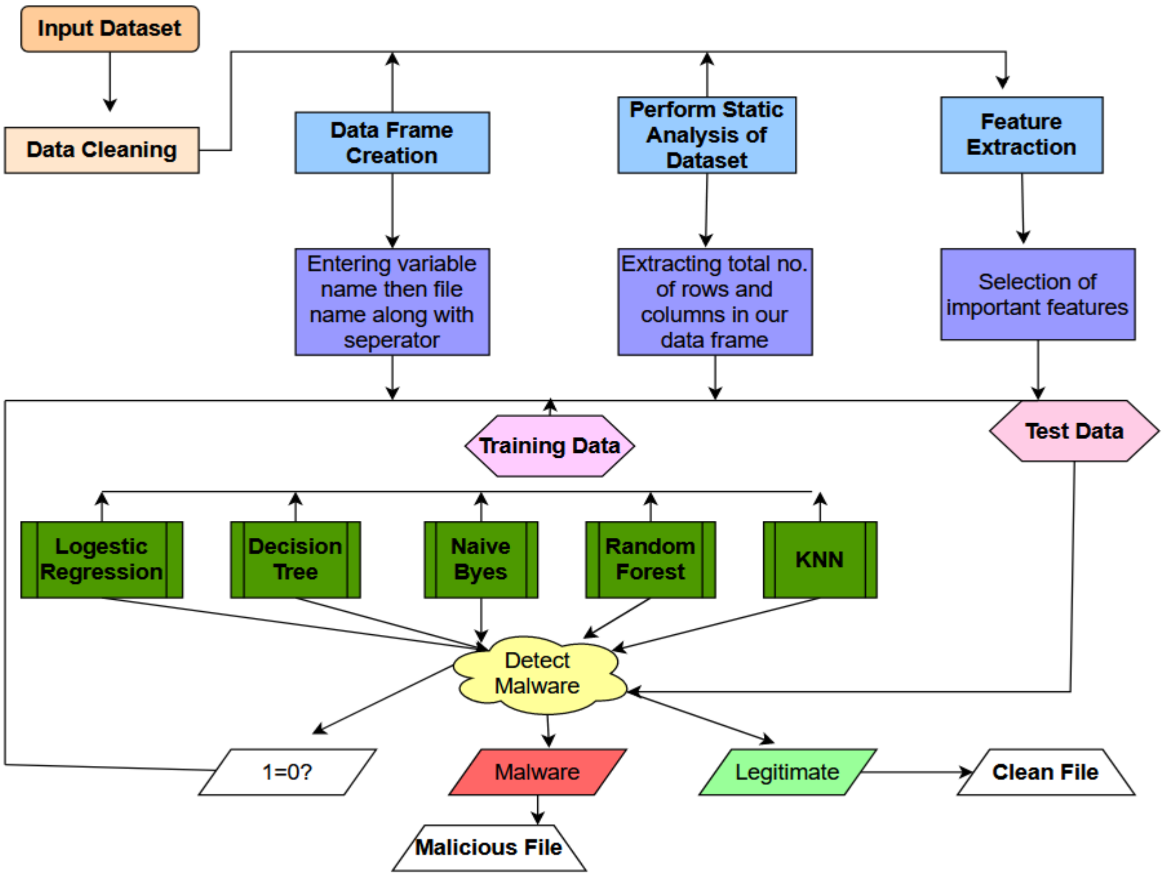
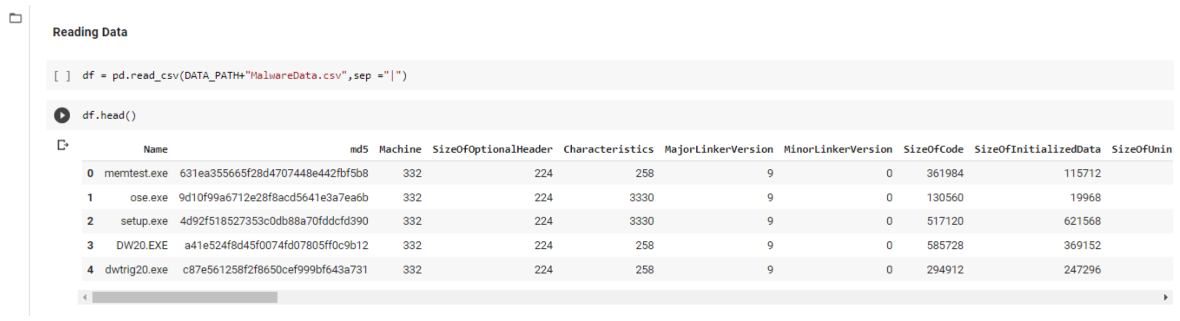


Figure 4: Proposed Method

5 Implementation

For Implementation we have used google colab platform which is freely available and providing computing resource for implementing various deep learning and machine learning algorithms. We have first uploaded our two datasets in google drive and then drive is mounted using google authenticator.



Reading Data

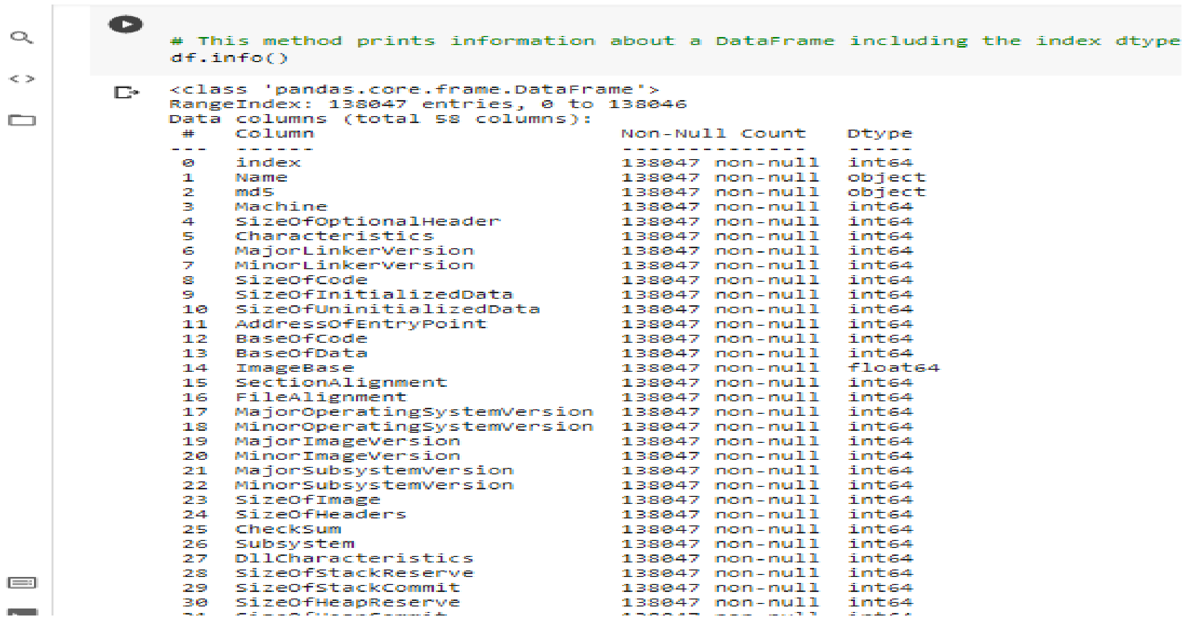
```
[ ] df = pd.read_csv(DATA_PATH+"MalwareData.csv",sep ="|")
```

```
df.head()
```

	Name	md5	Machine	SizeOfOptionalHeader	Characteristics	MajorLinkerVersion	MinorLinkerVersion	SizeOfCode	SizeOfInitializedData	SizeOfUnin
0	memtest.exe	631ea355665f28d4707448e442fbf5b8	332	224	258	9	0	361984	115712	
1	ose.exe	9d10f99a6712e28f8acd5641e3a7ea6b	332	224	3330	9	0	130560	19968	
2	setup.exe	4d92f518527353c0db88a70fddcfd390	332	224	3330	9	0	517120	621568	
3	DW20.EXE	a41e524f8d45f0074fd07805ff0c9b12	332	224	258	9	0	585728	369152	
4	dwtrig20.exe	c87e561258f2f8650cef999bf643a731	332	224	258	9	0	294912	247296	

Figure 5: Reading Virus Share Dataset

In above figure we can see that we have created a dataframe in which we have added our virusshare dataset.



```
# This method prints information about a DataFrame including the index dtype
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 138047 entries, 0 to 138046
Data columns (total 58 columns):
 #   Column                         Non-Null Count   Dtype
---  ------                         --------------   -----
 0   index                          138047 non-null  int64
 1   Name                           138047 non-null  object
 2   md5                            138047 non-null  object
 3   Machine                        138047 non-null  int64
 4   SizeOfOptionalHeader           138047 non-null  int64
 5   Characteristics                138047 non-null  int64
 6   MajorLinkerVersion             138047 non-null  int64
 7   MinorLinkerVersion             138047 non-null  int64
 8   SizeOfCode                     138047 non-null  int64
 9   SizeOfInitializedData          138047 non-null  int64
 10  SizeOfUninitializedData        138047 non-null  int64
 11  AddressOfEntryPoint            138047 non-null  int64
 12  BaseOfCode                     138047 non-null  int64
 13  BaseOfData                     138047 non-null  int64
 14  ImageBase                      138047 non-null  float64
 15  SectionAlignment               138047 non-null  int64
 16  FileAlignment                  138047 non-null  int64
 17  MajorOperatingSystemVersion    138047 non-null  int64
 18  MinorOperatingSystemVersion    138047 non-null  int64
 19  MajorImageVersion              138047 non-null  int64
 20  MinorImageVersion              138047 non-null  int64
 21  MajorSubsystemVersion          138047 non-null  int64
 22  MinorSubsystemVersion          138047 non-null  int64
 23  SizeOfImage                    138047 non-null  int64
 24  SizeOfHeaders                  138047 non-null  int64
 25  CheckSum                       138047 non-null  int64
 26  Subsystem                      138047 non-null  int64
 27  DllCharacteristics             138047 non-null  int64
 28  SizeOfStackReserve             138047 non-null  int64
 29  SizeOfStackCommit              138047 non-null  int64
 30  SizeOfHeapReserve              138047 non-null  int64
```

Figure 6: Data Cleaning

In above figure we have checked for any null values in our features so we can clean dataset and we found there are no null values in any feature. If null values or data cleaning will not done than some data overfitting and underfitting issues may reside. In our dataset there were total 58 features and all are checked against null values.

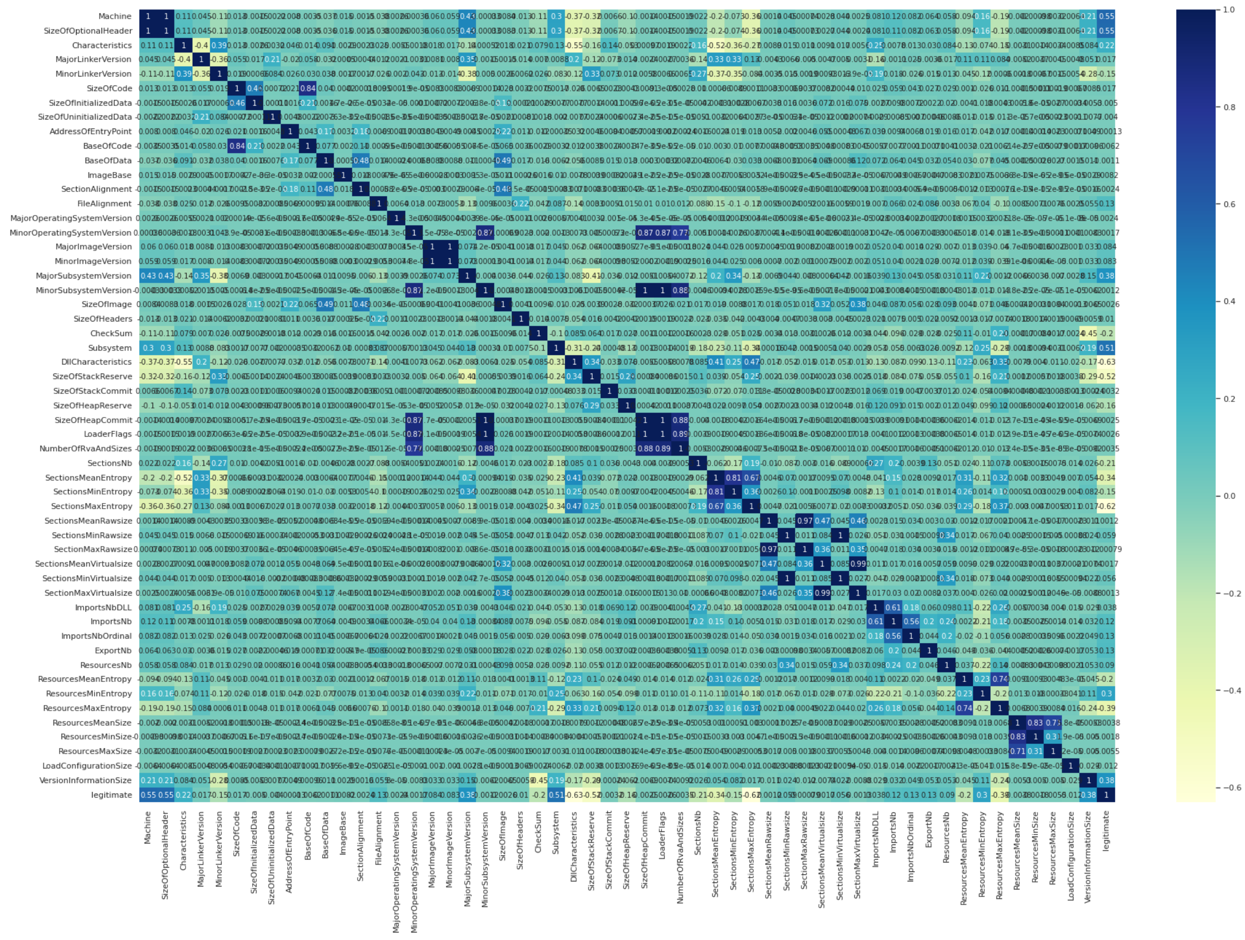


Figure 7: Feature correlation

Correlation is a proportion of the straight relationship of at least 2 factors. Through relationship, we can foresee one variable from the other. The rationale behind utilizing relationship for include choice is that the acceptable factors are profoundly corresponded with the objective. Moreover, factors ought to be corresponded with the objective yet ought to be uncorrelated among themselves. If two factors are associated, we can foresee one from the other. Along these lines, if two highlights are connected, the model just actually needs one of them, as the subsequent one doesn't add extra data. We have utilized the Pearson Correlation here.

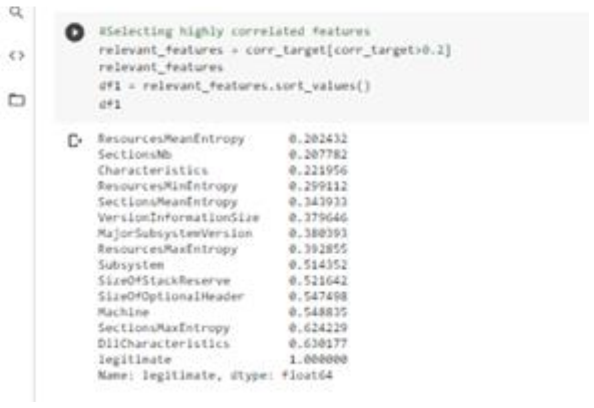
In above figure we have mapped all features using pearson correlation and checked whether any partial distribution of data occurs or not.

['Name', 'md5', 'Machine', 'SizeOfOptionalHeader', 'Characteristics', 'MajorLinkerVersion', 'MinorLinkerVersion', 'SizeOfCode', 'SizeOfInitializedData', 'SizeOfUninitializedData', 'AddressOfEntryPoint', 'BaseOfCode', 'BaseOfData', 'ImageBase', 'SectionAlignment', 'FileAlignment', 'MajorOperatingSystemVersion', 'MinorOperatingSystemVersion', 'MajorImageVersion', 'MinorImageVersion', 'MajorSubsystemVersion', 'MinorSubsystemVersion', 'SizeOfImage',

'SizeOfHeaders', 'CheckSum', 'Subsystem', 'DllCharacteristics',
'SizeOfStackReserve', 'SizeOfStackCommit', 'SizeOfHeapReserve',
'SizeOfHeapCommit', 'LoaderFlags', 'NumberOfRvaAndSizes', 'SectionsNb',
'SectionsMeanEntropy', 'SectionsMinEntropy', 'SectionsMaxEntropy',
'SectionsMeanRawsize', 'SectionsMinRawsize', 'SectionMaxRawsize',
'SectionsMeanVirtualsize', 'SectionsMinVirtualsize',
'SectionMaxVirtualsize', 'ImportsNbDLL', 'ImportsNb',
'ImportsNbOrdinal', 'ExportNb', 'ResourcesNb', 'ResourcesMeanEntropy',
'ResourcesMinEntropy', 'ResourcesMaxEntropy', 'ResourcesMeanSize',
'ResourcesMinSize', 'ResourcesMaxSize', 'LoadConfigurationSize',
'VersionInformationSize', 'legitimate']

The above list is 58 features of our virusshare dataset has been displayed from these features we will select only relevant and important features for training and testing of our models.

Feature extraction can be used to characterized as changing the huge, ambiguous assortment of contributions to the arrangement of highlights. Progressed identification mainly depends on highlighting of extraction of the malicious files being examined. Feature could contain various plaintext strings found in the dismantled documents, the size of the malware, n-gram byte arrangements, framework asset data like the arrangement of DLLs, and so forth by utilizing AI calculation, these highlights are given as sources of info.



```
#Selecting highly correlated features
relevant_features = corr_target[corr_target>0.2]
relevant_features
df1 = relevant_features.sort_values()
df1
```

```
ResourcesMeanEntropy      0.202432
SectionsNb                0.207782
Characteristics           0.221956
ResourcesMinEntropy       0.299112
SectionsMeanEntropy       0.343933
VersionInformationSize    0.379646
MajorSubsystemVersion     0.380393
ResourcesMaxEntropy       0.392855
Subsystem                 0.514352
SizeOfStackReserve        0.521642
SizeOfOptionalHeader      0.547498
Machine                   0.548835
SectionsMaxEntropy        0.624229
DllCharacteristics        0.630177
legitimate                1.000000
Name: legitimate, dtype: float64
```

Figure 7: Feature Selection

We have used coefficient to find important features and used the features in which coefficient is grater than 0.2 for training and testing and other features were not used as only important features have been selected. From this method we have identified 14 important features out of 58 features. In below figure we have checked the distribution of values in 14 important features.

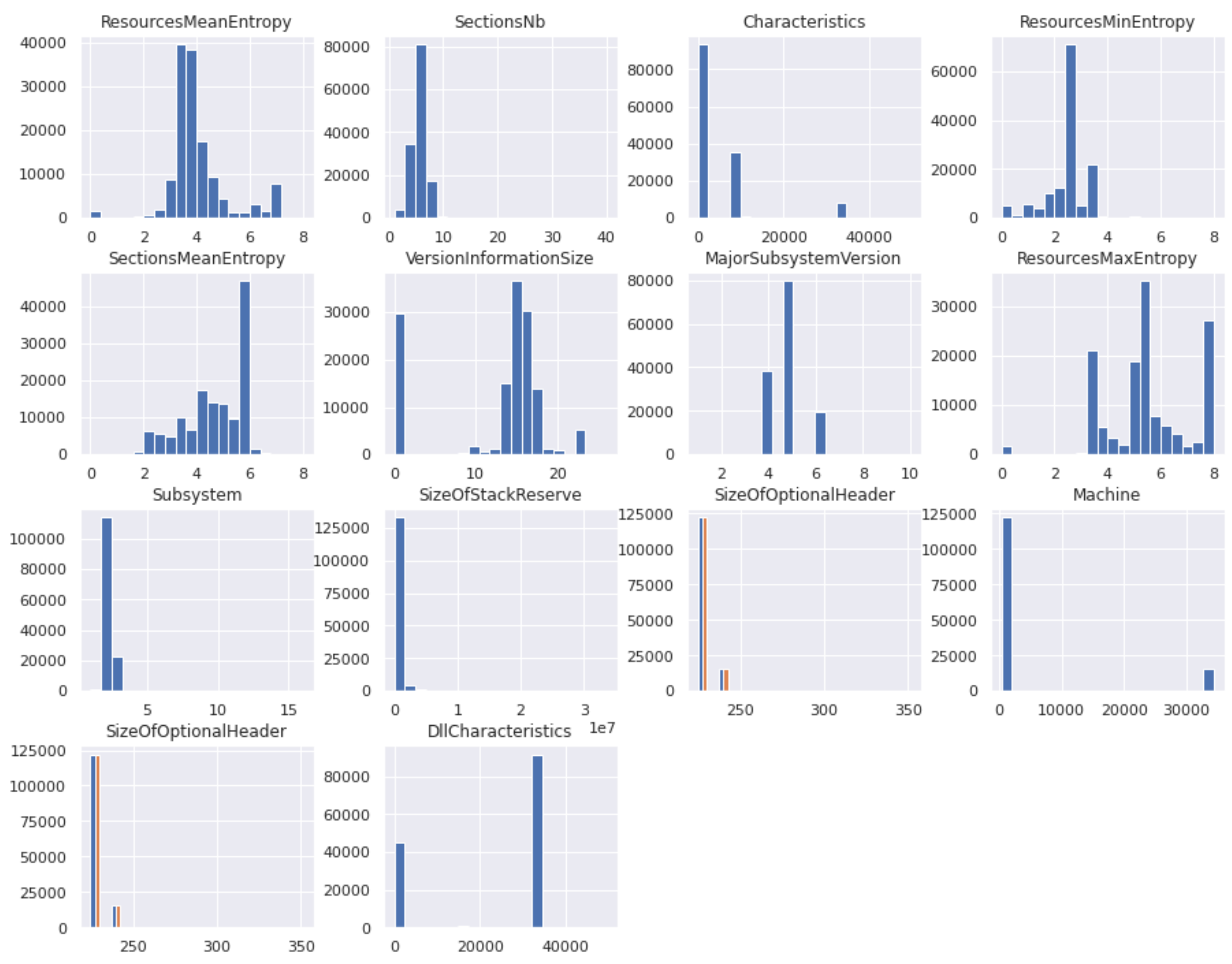
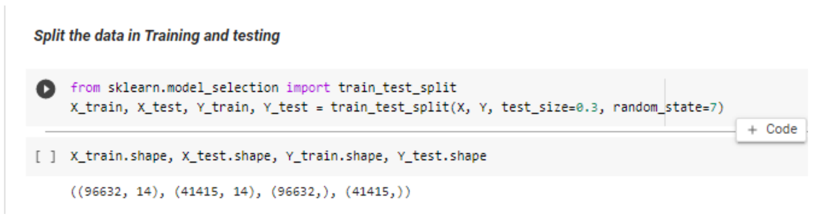


Figure 8: Values distribution of important features



Split the data in Training and testing

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=7)
```

```
X_train.shape, X_test.shape, Y_train.shape, Y_test.shape
```

```
((96632, 14), (41415, 14), (96632,), (41415,))
```

Figure 9: Training and Testing Data Split

In above figure we have split the data for training and testing in 70/30 using sklearn library function. Here we found we have 966632 samples for training and 41415 random samples for testing.

We have used scikit leran library function for LogisticRegression, DecisionTreeClassifier, RandomForestClassifier, GaussianNB and KNeighborsClassifier. First these models are trained using model evaluate function and we have used ML-Ensemble library.

6 Evaluation

We have trained five different machine learning models and checked the performance and testing accuracy of each model on virus share and UCI dataset. The result gained from each model have been displayed here.

6.1 Virus Share Dataset

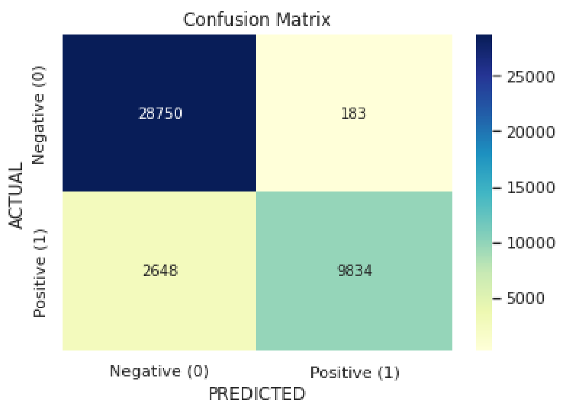


Figure 10: CF Matrix of LR

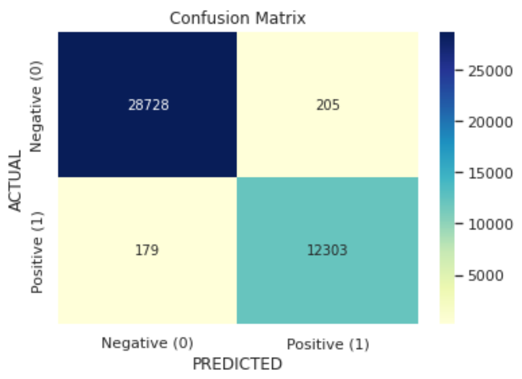


Figure 11: CF Matrix of DT

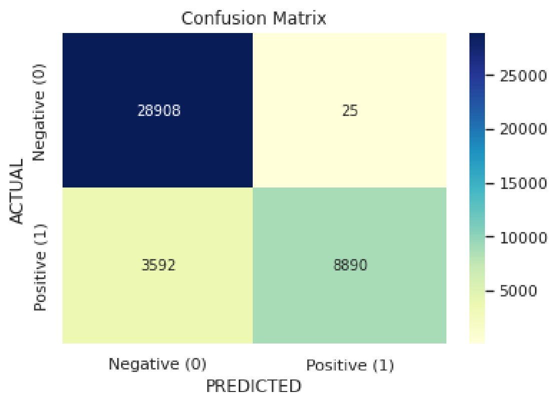


Figure 12: CF Matrix of NB

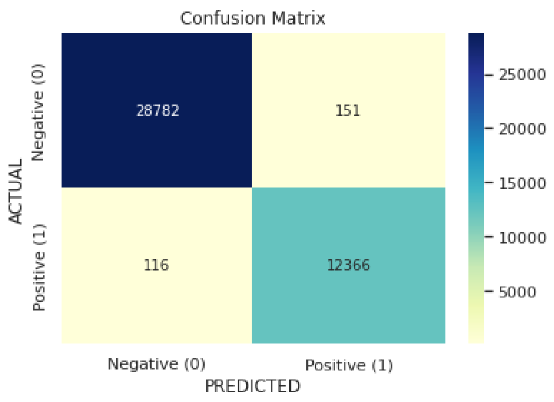


Figure 13: CF Matrix of RF

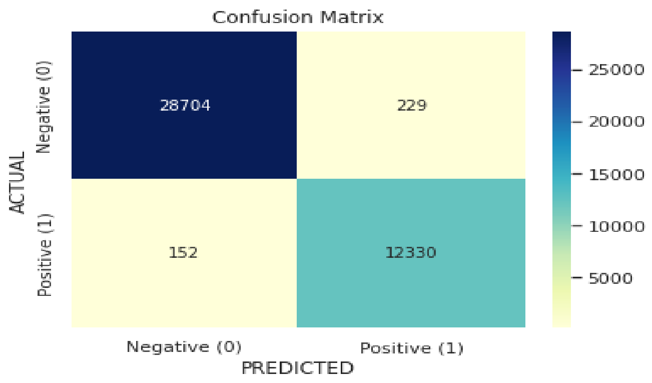
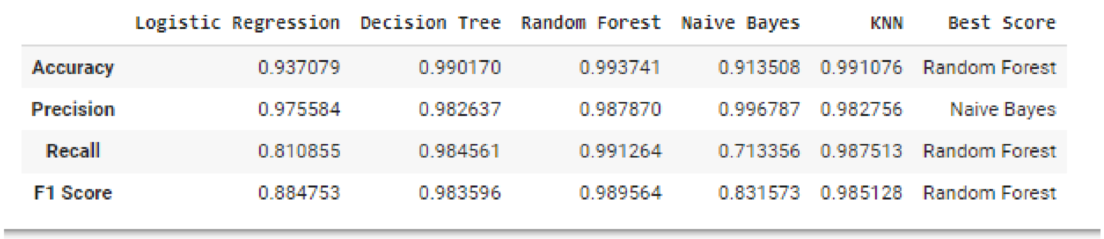


Figure 12: CF Matrix of KNN

In above figures confusion matrix of each model are displayed and based on True Positive, True Negative, False Positive and False Negative confusion matrix resulted from which accuracy, precision, recall can be calculated.

Table 2: ML Model Result Comparison



	Logistic Regression	Decision Tree	Random Forest	Naive Bayes	KNN	Best Score
Accuracy	0.937079	0.990170	0.993741	0.913508	0.991076	Random Forest
Precision	0.975584	0.982637	0.987870	0.996787	0.982756	Naive Bayes
Recall	0.810855	0.984561	0.991264	0.713356	0.987513	Random Forest
F1 Score	0.884753	0.983596	0.989564	0.831573	0.985128	Random Forest

In above table best score in accuracy, recall and F1 Score has been achived in Random Forest. Only Precision is good in Naïve Bayes. Random Forest provides best result in accuracy and F1 Score in terms of different dataset.

In figure 13 we can see that testing accuracy of Random forest model remains high than all other ML models and provides good testing results.

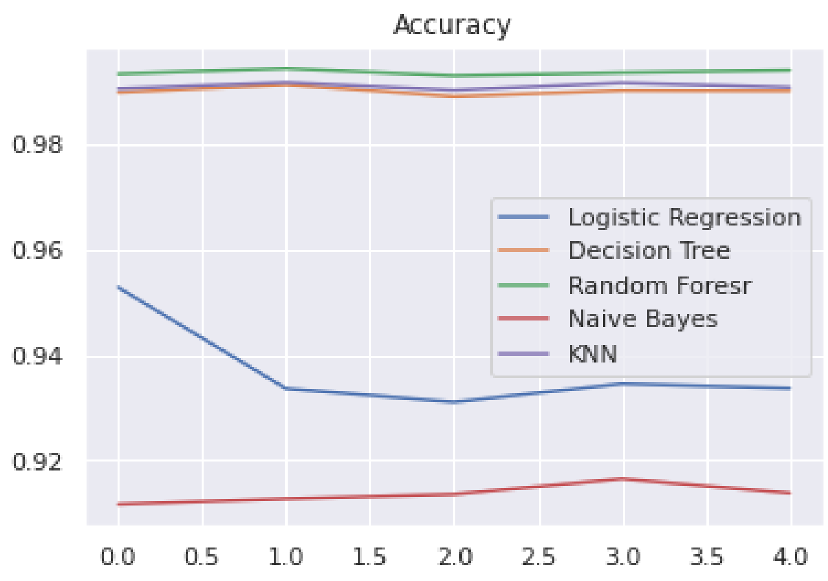


Figure 13: Accuracy of Testing of ML Models

Probablistic predictions of binary classification can be compared by calibration curves. For binned predictions, calibration curve used to plot true frequency of true lable on its predicted probability. In below figure we have given calibration plot of all five ML models. In which random forest gives highest fraction of positives values.

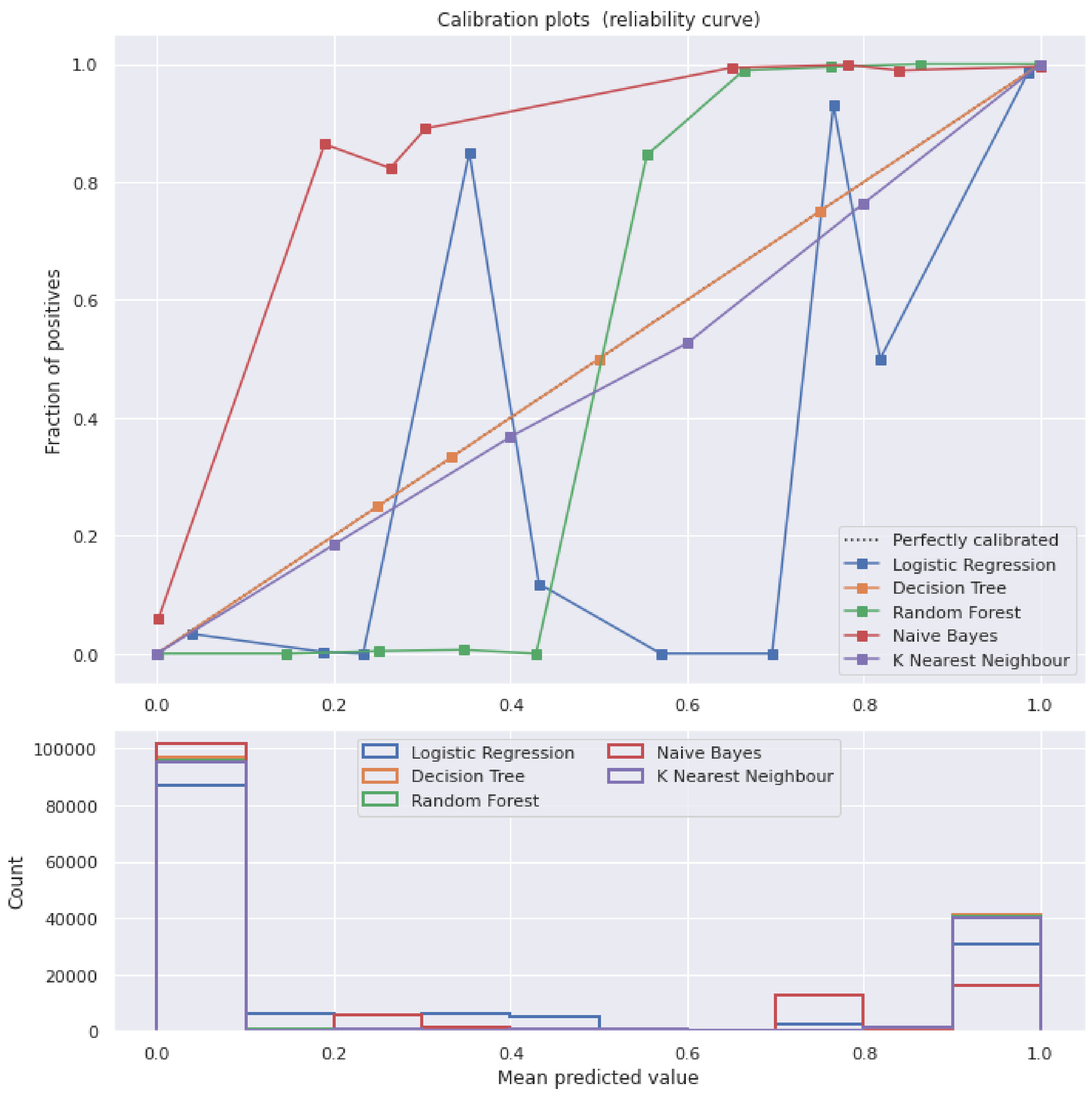


Figure 13: Calibration Curve of ML Models

6.2 UCI Dataset

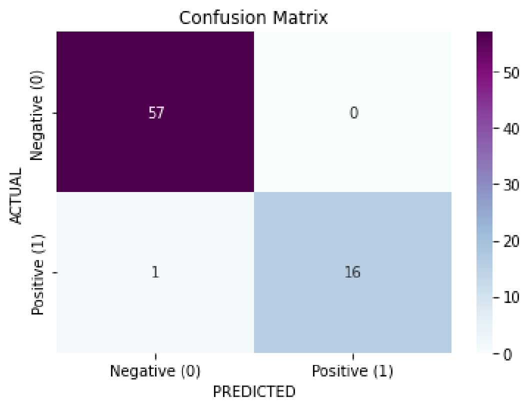


Figure 14: CF Matrix of LR

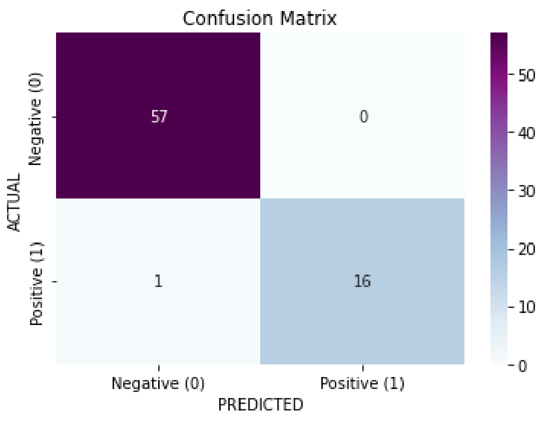


Figure 15: CF Matrix of DT

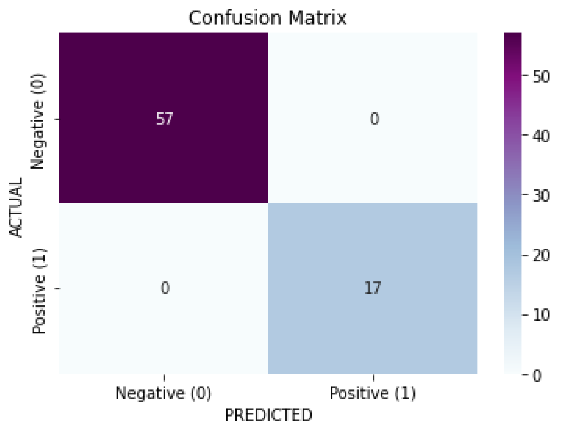


Figure 16: CF Matrix of NB

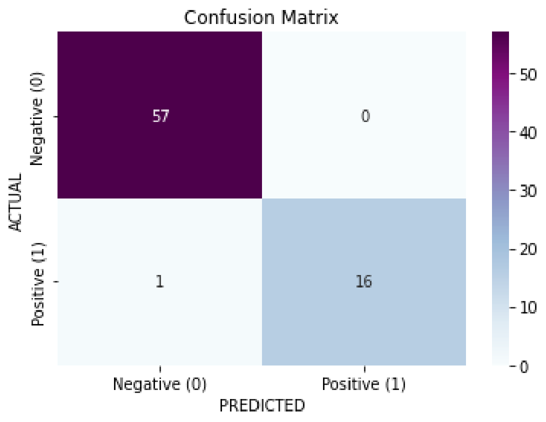


Figure 17: CF Matrix of RF

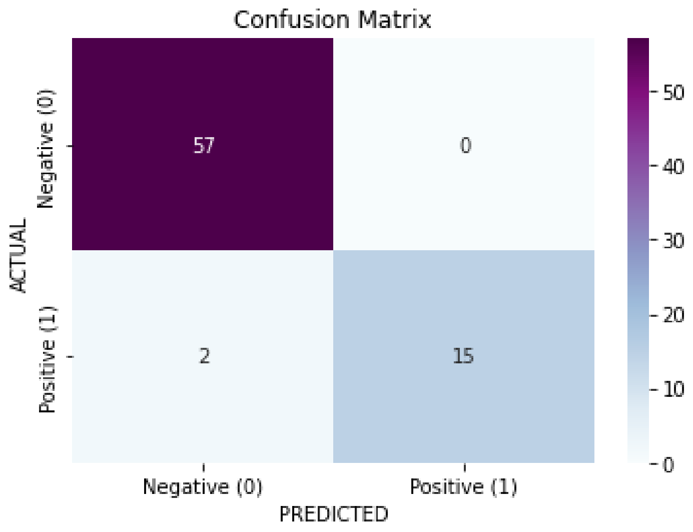
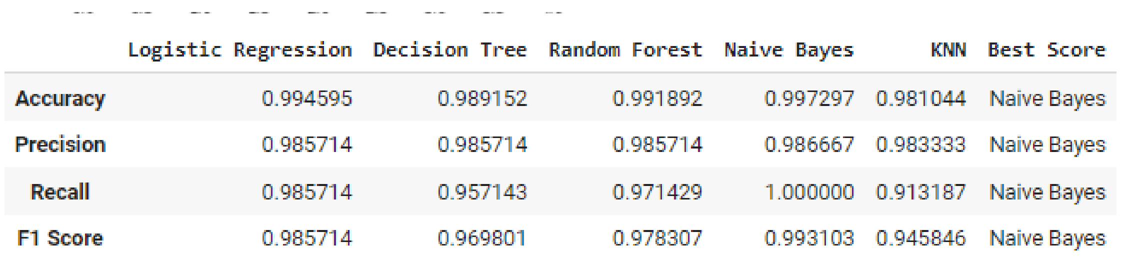


Figure 18: CF Matrix of KNN

In above figure we have displayed different confusion matrix of all five ML models on UCI malware dataset which is taken from Kaggle, we can see that in this dataset Naïve Bayes provide lowest rate of False Positive and false negative and KNN provide highest rate of false positive and false negative.

Table 3: ML Model Result Comparison (UCI Malware Dataset)



	Logistic Regression	Decision Tree	Random Forest	Naive Bayes	KNN	Best Score
Accuracy	0.994595	0.989152	0.991892	0.997297	0.981044	Naive Bayes
Precision	0.985714	0.985714	0.985714	0.986667	0.983333	Naive Bayes
Recall	0.985714	0.957143	0.971429	1.000000	0.913187	Naive Bayes
F1 Score	0.985714	0.969801	0.978307	0.993103	0.945846	Naive Bayes

In above table best score in accuracy, precision,recall and F1 Score has been achived in Naïve bayes. Naïve bayes provides good result than random forest in this dataset.

In figure 19 we can see that testing accuracy of naïve bayes model remains high than all other ML models and provides good testing results.

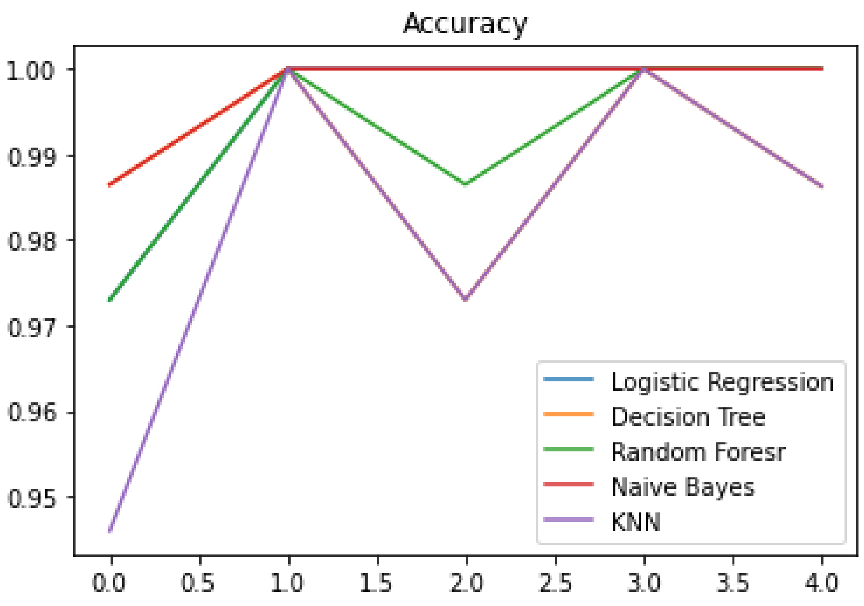


Figure 19: Accuracy of Testing of ML Models

In figure 20 we have displayed calibration plots of all five models on UCI dataset.

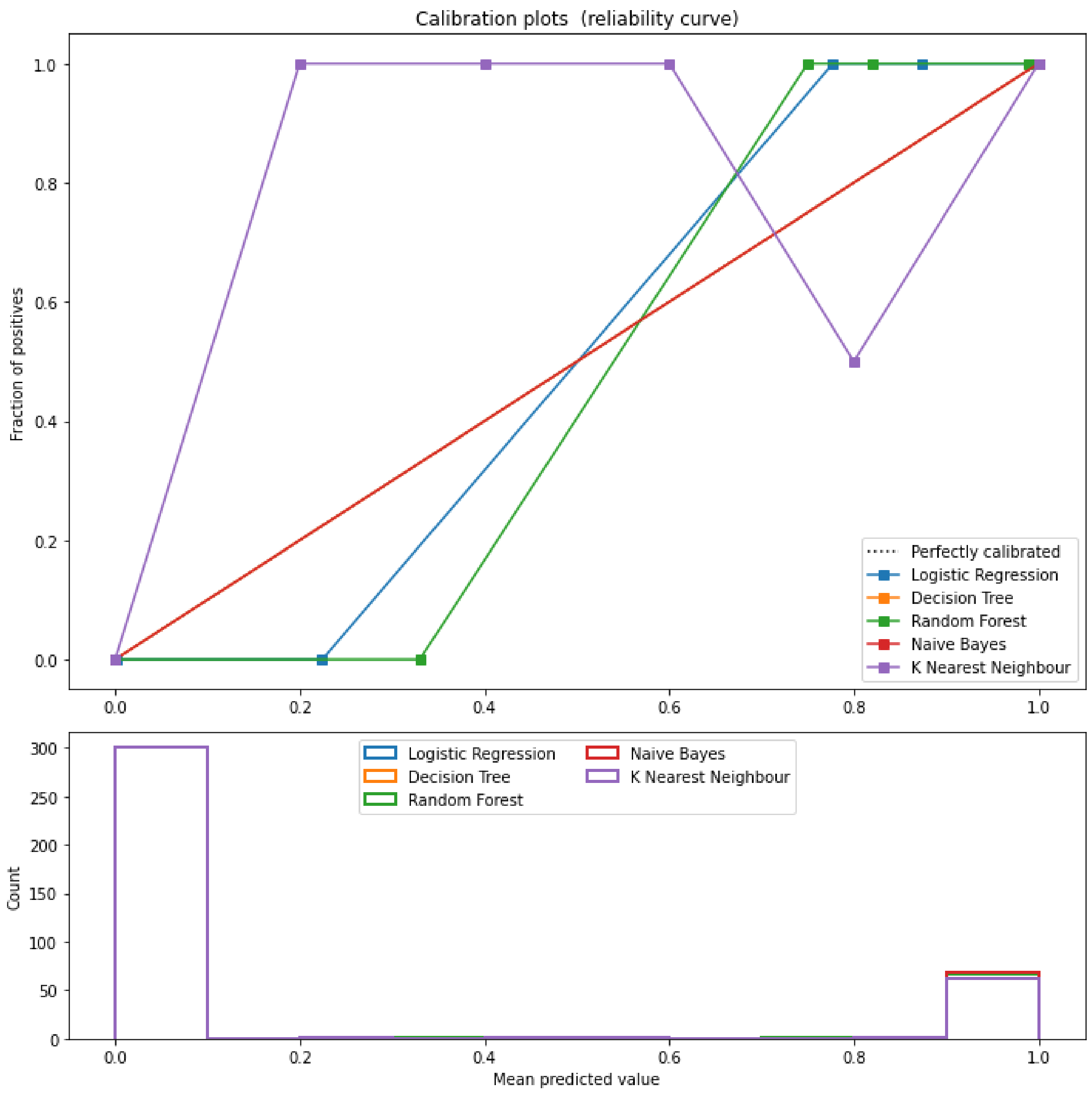


Figure 20: Calibration Curve of ML Models

From above figure we can see that on uci dataset only Naive Bayes is perfectly calibrated.
Calibration curves are used to evaluate how calibrated a classifier is i.e., how the probabilities of predicting each class label differ. The x-axis represents the average predicted probability in each bin. The y-axis is the ratio of positives (the proportion of positive predictions). The curve of the ideal calibrated model is a linear straight line from (0, 0) moving linearly.
We have compared Probablistic predictions of binary classification of UCI dataset. Probablistic predictions of binary classification can be compared by calibration curves. For binned predictions, calibration curve used to plot true frequency of true lable on its predicted probability. In above figure we have given calibration plot of all five ML models. In which Naïve Bayes gives highest fraction of positives values. In Virus-Share dataset (Figure 13) we got highest fraction of positives values in Random Forest.

6.3 Discussion

Previous literature research on malware detection demonstrate that categorization may be done successfully with the aid of machine learning approaches, although several challenges remain unresolved. Zero-day attacks are those that occur on a day when no new malware is released.
It is the primary goal of all malware researchers. Some concerns and obstacles with malware detection remain and have yet to be overcome. When it comes to reality, one challenge is that real or manual verification of categorization findings becomes more difficult. Another difficulty is how to progress strategies for more active learning. More recent developments in machine learning, ensemble learning, deep learning, and other domains are expected. Zero-day attacks necessitate the use of the most modern tactics. One of the problems is dealing with huge datasets. These sophisticated strategies are required in the decrease of dimensionality.

This study focused on malware detection and evaluated the false-positive rate of detection using supervised and un-supervised machine learning algorithms. In former model [14] proposed a discovery of malicious files which is dependent on API calls and their contentions. In other work [15] they utilized a novel two sorts of classifiers which are Logic Model tree and K-Means calculations. Further [18] The creators generally utilize the static highlights acquired by against aggregation of APK. These static highlights have incorporated string, touchy API, declarations and application authorizations. Later [27] Image processing techniques has been used for classification and visualization of malware.

In contrast to comparable research, our model detected Malware with a 99.1% recall score, whereas [14] found 98.4% and [15] discovered 97.68%. The primary model might produce a list of the most significant API, Permission, and Network properties. These chosen significant qualities are comparable to those chosen by [14], [15], [16], [18], and [27]. However, there are modest discrepancies that might be attributed to the quantity of samples used in research.

7 Conclusion and Future Work

We trained and tested several machine learning models on two datasets and all models provide very reasonable results. On Virus share dataset Random Forest provides highest accuracy 99.37% and naïve bayes provides 91.30% accuracy in this dataset. On UCI dataset naïve bayes provides highest accuracy 99.72%. We can also apply deep learning models on this dataset and gain some more accuracy. Future work we will try to build new dataset from recent malware and try to provide some application to check whether the files are malicious or not. Here we have used 14 important features based on pearson corelation coefficient and trained and tested the model. We will also try to use optuna hypervisor for identifying best parameters in ML models so we can improve the accuracy.

8 References

[1] Apvrille, “Digsig: Run-time authentication of binaries at kernel level,” Proceedings of LISA, Jan. 2004, Accessed: Dec. 13, 2021. [Online]. Available: https://www.academia.edu/17245897/Digsig_Run_time_authentication_of_binaries_at_kernel_level.

[2] “Limits of Static Analysis for Malware Detection | SE,” *se.fbk.eu*. https://se.fbk.eu/events/limits-static-analysis-malware-detection (accessed Dec. 13, 2021).

[3] G. Jacob, H. Debar, and E. Filiol, “Behavioral detection of malware: from a survey towards an established taxonomy,” *Journal in Computer Virology*, vol. 4, no. 3, pp. 251–266, Feb. 2008, doi: 10.1007/s11416-008-0086-0.

[4] D. R. Ellis, J. G. Aiken, K. S. Attwood, and S. D. Tenaglia, “A behavioral approach to worm detection,” *Proceedings of the 2004 ACM workshop on Rapid malcode - WORM ’04*, 2004, doi: 10.1145/1029618.1029625.

[5] P. A. Porras and R. A. Kemmerer, “Penetration state transition analysis: A rule-based intrusion detection approach,” *IEEE Xplore*, Nov. 01, 1992. https://ieeexplore.ieee.org/document/228217 (accessed Dec. 13, 2021).

[6] “Annual Worldwide Economic Damages from Malware Exceed $13 Billion | Computer Economics -- for IT metrics, ratios, benchmarks, and research advisories for IT management,” *Computereconomics.com*, 2019. https://www.computereconomics.com/article.cfm?id=1225.

[7] “Cybercrime To Cost The World $10.5 Trillion Annually By 2025,” *Cybercrime Magazine*, Feb. 21, 2018. https://cybersecurityventures.com/hackerpocalypse-cybercrime-report-2016. (accessed Dec. 13, 2021).

[8] Mohammad DK, Mohd TS, Rafia A, Mahenoor S,& Sonalii S” Malware detection using Machine Learning Algorithms” IJARCCE, Vol. 6, Issue 9, September (2017), available online: https://ijarcce.com/upload/2017/september17/IJARCCE%2035.pdf.

[9] M. IMRAN, M. T. AFZAL, and M. A. QADIR, “A comparison of feature extraction techniques for malware analysis,” *TURKISH JOURNAL OF ELECTRICAL ENGINEERING & COMPUTER SCIENCES*, vol. 25, pp. 1173–1183, 2017, doi: 10.3906/elk-1601-189.

[10] J. Landage and P. M. P. Wankhade, “Malware and Malware Detection Techniques : A Survey,” *International Journal of Engineering Research & Technology*, vol. 2, no. 12, Dec. 2013, Accessed: Dec. 13, 2021. [Online]. Available: https://www.ijert.org/malware-and-malware-detection-techniques-a-survey-2.

[11] S. Najari, “Malware Detection Using Data Mining Techniques,” *International Journal of Intelligent Information Systems*, vol. 3, no. 6, p. 33, 2014, doi: 10.11648/j.ijiis.s.2014030601.16.

[12] N. Hande and nbspProf V. Rao, “A comparative study of static, dynamic and hybrid analysis techniques for android malware detection,” *undefined*, 2017, Accessed: Dec. 13, 2021. [Online]. Available: https://www.semanticscholar.org/paper/A-comparative-study-of-static%2C-dynamic-and-hybrid-Hande-Rao/dd1ef3cd014499ede460d90f109ac93cd694c726.

[13] R. Tian, R. Islam, L. Batten, and S. Versteeg, “Differentiating malware from cleanware using behavioural analysis,” *IEEE Xplore*, Oct. 01, 2010. https://ieeexplore.ieee.org/document/5665796 (accessed Dec. 13, 2021).

[14] Z. Salehi, M. Ghiasi, and A. Sami, “A miner for malware detection based on API function calls and their arguments,” *IEEE Xplore*, May 01, 2012. https://ieeexplore.ieee.org/document/6313810 (accessed Dec. 13, 2021).

[15] A. Dhammi and M. Singh, “Behavior analysis of malware using machine learning,” *IEEE Xplore*, Aug. 01, 2015. https://ieeexplore.ieee.org/document/7346730.

[16] P. V. Shijo and A. Salim, “Integrated Static and Dynamic Analysis for Malware Detection,” *Procedia Computer Science*, vol. 46, pp. 804–811, 2015, doi: 10.1016/j.procs.2015.02.149.

[17] M. Shaiful, A. Bin, M. Sari, M. Maarof, and T. Malaysia, “Classification of Malware Family Using Decision Tree Algorithm.” Accessed: Dec. 13, 2021. [Online]. Available: https://engineering.utm.my/computing/proceeding/wp-content/uploads/sites/114/2018/04/Classification-of-Malware-Family-Using-Decision-Tree-Algorithm.pdf.

[18] L. Yang, “Employing The Algorithms Of Random Forest And Neural Networks For The Detection And Analysis Of Malicious Code Of Android Applications,” Beijing Jiaotong University, 2015.

[19] I. Santos, F. Brezo, X. Ugarte-Pedrero, and P. G. Bringas, “Opcode sequences as representation of executables for data-mining-based unknown malware detection,” *Information Sciences*, vol. 231, pp. 64–82, May 2013, doi: 10.1016/j.ins.2011.08.020.

[20] C.-I. Fan, H.-W. Hsiao, C.-H. Chou, and Y.-F. Tseng, “Malware Detection Systems Based on API Log Data Mining,” *2015 IEEE 39th Annual Computer Software and Applications Conference*, Jul. 2015, doi: 10.1109/compsac.2015.241.

[21] M. Belaoued and S. Mazouzi, “A Real-Time PE-Malware Detection System Based on CHI-Square Test and PE-File Features,” *IFIP Advances in Information and Communication Technology*, pp. 416–425, 2015, doi: 10.1007/978-3-319-19578-0_34.

[22] M. Hassen, M. M. Carvalho, and P. K. Chan, “Malware classification using static analysis based features,” *IEEE Xplore*, Nov. 01, 2017. https://ieeexplore.ieee.org/abstract/document/8285426 (accessed Dec. 13, 2021).

[23] M. Christodorescu, S. Jha, and C. Kruegel, “Mining specifications of malicious behavior,” *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering - ESEC-FSE ’07*, 2007, doi: 10.1145/1287624.1287628.

[24] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, “DL-Droid: Deep learning based android malware detection using real devices,” *Computers & Security*, vol. 89, p. 101663, Feb. 2020, doi: 10.1016/j.cose.2019.101663.

[25] G. Canfora, E. Medvet, F. Mercaldo, and C. A. Visaggio, “Detecting Android malware using sequences of system calls,” *Proceedings of the 3rd International Workshop on Software Development Lifecycle for Mobile*, Aug. 2015, doi: 10.1145/2804345.2804349.

[26] W. Wang, Y. Li, X. Wang, J. Liu, and X. Zhang, “Detecting Android malicious apps and categorizing benign apps with ensemble of classifiers,” *Future Generation Computer Systems*, vol. 78, pp. 987–994, Jan. 2018, doi: 10.1016/j.future.2017.01.019.

[27] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, “Malware images,” *Proceedings of the 8th International Symposium on Visualization for Cyber Security - VizSec ’11*, 2011, doi: 10.1145/2016904.2016908.

[28] Corvus Forensics. VirusShare. url: https://virusshare.com. Accessed: 01.02.2021.