**A PROJECT REPORT**

**ON**

**MALWARE DETECTION USING MACHINE**

**LEARNING AND DEEP LEARNING**

**ALGORITHMS**

***Submitted by***

**AAKASH KUMAR (40327203118)**

**AMAN CHAUDHARY (44727203118)**

**ARNAV MALHOTRA (42627203118)**

**HIMANSHU (44827203118)**

***Under the Guidance of***

**Ms. TANNU VATS**

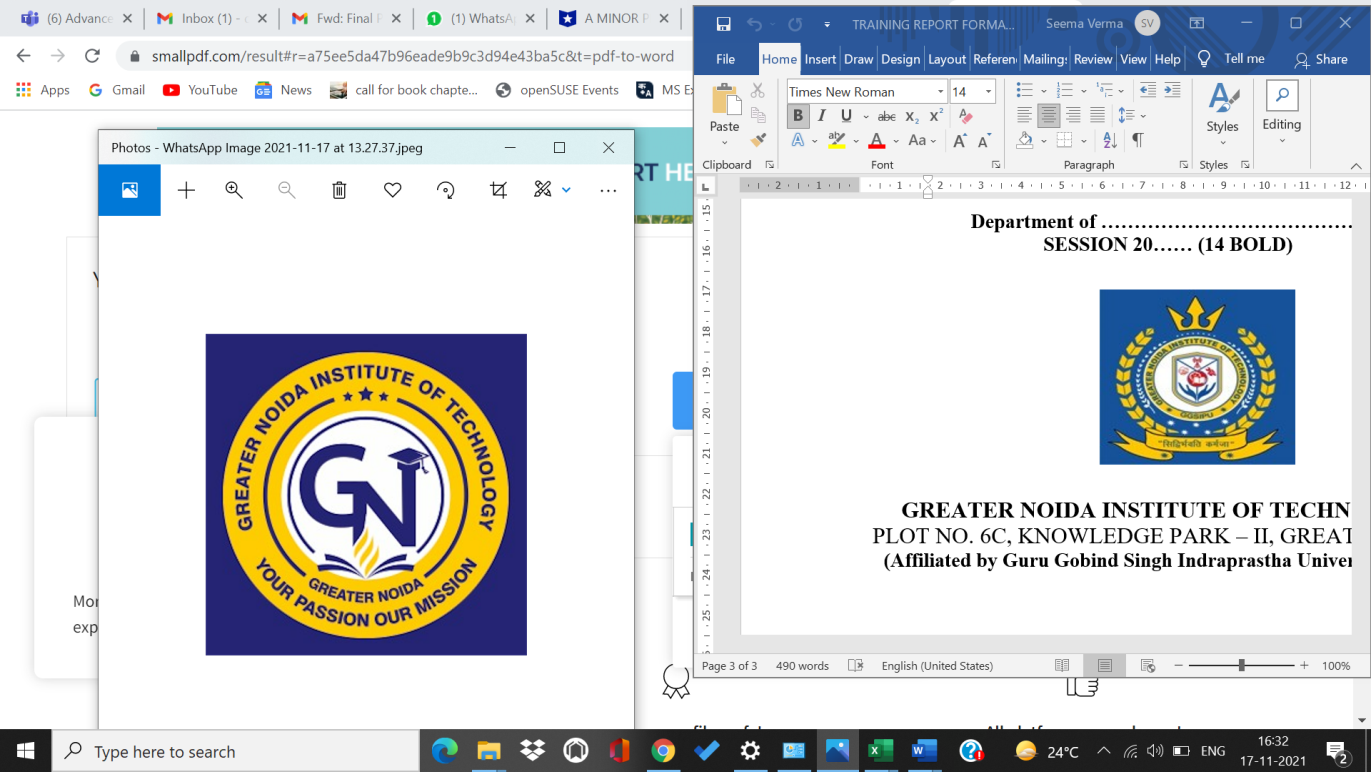
**ASST. PROF, DEPT. OF IT**

***in partial fulfillment for the award of the degree of***

**BACHELOR OF TECHNOLOGY**

**IN**

**INFORMATION TECHNOLOGY**



**GREATER NOIDA INSTITUTE OF TECHNOLOGY**

**PLOT NO. 6C, KNOWLEDGE PARK – II, GREATER NOIDA**

**(Affiliated by Guru Gobind Singh Indraprastha University)**

**JUNE, 2022**

**Declaration**

We hereby declare that this project report entitled “**MALWARE DETECTION USING MACHINE LEARNING AND DEEP LEARNING ALGORITHMS**” *by* **AAKASH KUMAR(40327203118) ,AMAN CHAUDHARY(44727203118),ARNAV MALHOTRA (42627203118) and HIMANSHU(44827203118),** being submitted in partial fulfillment of the requirements for the degree of Bachelor of Technology in **Information Technology** , GNIT(IPU), Greater Noida, during the academic year **2018-2022**, is a bonafide record of our original work carried out under the guidance of **Ms. TANNU VATS , ASST. PROF. , DEPT. OF IT .**

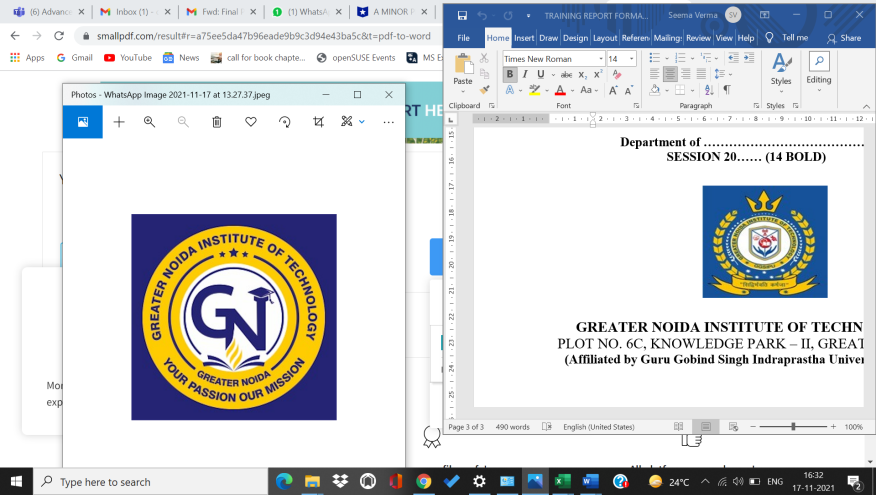
We further declare that we have not submitted the matter presented in this Project for the award of any other Degree/Diploma of this University or any other University/Institute.

1. AAKASH KUMAR, 40327203118

2. AMAN CHAUDHARY, 44727203118

3. ARNAV MALHOTRA, 42627203118

4. HIMANSHU, 44827203118

**GREATER NOIDA INSTITUTE OF TECHNOLOGY**

PLOT NO. 6C, KNOWLEDGE PARK – II, GREATER NOIDA

(Affiliated by Guru Gobind Singh Indraprastha University)

**Department of Information Technology**

JUNE, 2022

**Certificate**

This is to certify that this project report entitled “**MALWARE DETECTION USING MACHINE LEARNING AND DEEP LEARNING ALGORITHMS**” *by* **AAKASH KUMAR (40327203118), AMAN CHAUDHARY (44727203118), ARNAV MALHOTRA (42627203118), HIMANSHU(44827203118) ,** being**,** submitted in partial fulfillment of the requirements for the degree of Bachelor of Technology in **INFORMATION TECHNOLOGY** under GNIT, GGSIPU, Greater Noida, during the academic year **2018-2022**, is a bonafide record of work carried out under my guidance and supervision. We hereby declare that the work has been carried out under my supervision and has not been submitted elsewhere for any other purpose.

**(Signature of Project Guide)**

Ms. Tannu Vats

Asst. Professor

Department of Information Technology

GNIT, GGSIPU, Greater Noida

**(Signature of HoD)**

Mr. Achintya Pandey

Head of Department

Department. of Information Technology

GNIT, GGSIPU, Greater Noida

**ACKNOWLEDGEMENT**

The successful realization of project is an outgrowth of a consolidated effort of people from desperate fronts. We are thankful to **Ms. TANNU VATS** (Assistant Professor) for her variable advice and support extended to us without which we could not be able to complete our project for a success.

We express our deep gratitude to **Mr. ACHINTYA PANDEY,** Head of Department (IT) for his endless support and affection towards us. We would like to express our sincere gratitude to **Dr. MANOJ RANA**, Director, GNIT, IPU for constant encouragement that has helped us to inculcate the spirit of getting knowledge and dedication to the purpose.

We take this opportunity to express our sincere thanks to all staff members of **IT** department for the valuable suggestion and also to our family and friends for their support.

**ARNAV MALHOTRA**

42627203118

**HIMANSHU**

44827203118

**AMAN CHAUDHARY**

44727203118

**AAKASH KUMAR**

40327203118

**TABLE OF CONTENTS Page No.**

**Declaration**  i

**Certificate** ii

**Acknowledgement** iii

**Abstract** iv

**List of Tables** v

**List of Figures** vi

**1. INTRODUCTION** 10

1.1 Objective 10

1.2 Motivation 10

1.3 Background 11

1. **PROJECT DESCRIPTION AND GOALS** 14
   1. Project Description 14
      1. Data Collection 14
      2. Data Preprocessing 15
   2. Goals 16
2. **TECHNICAL SPECIFICATIONS** 17
   1. Hardware Specifications 17
   2. Software Specifications 17
   3. Modules Description 18
3. **DESIGN APPROACH AND DETAILS** 20
   1. Architecture 20
   2. Logistic Regression 21
   3. Random Forest 23
   4. Neural Networks 26
   5. Project Flow Diagram 30
4. **PROJECT DEMONSTRATION** 31
   1. Importing Modules and Data Cleaning 31
   2. Data Preprocessing 32
   3. Logistic Regression Implementation 34
   4. Random Forest Implementation 35
   5. Neural Network Implementation 36
5. **RESULTS** 36
   1. Logistic Regression 37
   2. Random Forest 37
   3. Neural Network 38
6. **CONCLUSION AND FUTURE WORK** 38
7. **REFERENCES** 39

**APPENDIX A** 40

### **List of Figures**

[**Figure No. Figure Caption Page No.** 1 Dataset Distribution 15](#_Toc35048)

[2 First Five rows of Dataset 15](#_Toc35049)

[3 Logistic Regression 21](#_Toc35050)

[4 Random Forest algorithm with differentdecision trees 23](#_Toc35051)

[5 Neural Network with all Layers 26](#_Toc35052)

[6 Nodal view of neural Network 26](#_Toc35053)

[7 A project flow diagram 30](#_Toc35054)

[8 Importing Modules 31](#_Toc35055)

[9 Importing Modules for Random Forest 31](#_Toc35056)

[10 Importing Modules for Neural Network 32](#_Toc35057)

[11 Cleaning Data 32](#_Toc35058)

[12 Data PreProcessing 33](#_Toc35059)

[13 Data Splitting 33](#_Toc35060)

[14 Logistic Regression Implementation 34](#_Toc35061)

[15 Importing Modules 35](#_Toc35062)

[16 Feature Selection for Random Forest 36](#_Toc35063)

[17 Implementation of Random Forest 36](#_Toc35064)

[18 Creating a model for neural Network 37](#_Toc35065)

[19 Epoch Training for neural Network 37](#_Toc35066)

[20 Accuracy of Logistic Regression 38](#_Toc35067)

[21 Confusion Matrix of Logistic Regression 38](#_Toc35068)

[22 Accuracy of Random Forest 39](#_Toc35069)

[23 Confusion Matrix of Random Forest 40](#_Toc35070)

24 Accuracy and f1 score of Neural Network 41

**List of Tables**

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Table Caption** | **Page No.** |
| 1 | Contribution by Members | 40 |

**ABSTRACT**

With fast turn of events and development of the web, malware is one of major digital dangers nowadays. Henceforth, malware detection is an important factor in the security of computer systems. Nowadays, attackers generally design polymeric malware; it is usually a type of malware that continuously changes its recognizable feature to fool detection techniques that uses typical signature based methods. That is why the need for Machine Learning based detection arises. In this work, behavioral-pattern may be achieved through static or dynamic analysis, afterward we can apply dissimilar ML and DL techniques to identify whether it's malware or not. We will be discussing mainly 3 techniques Random Forest, Logistic Regression and Neural Networks and we will compare the results obtained by each method.

Chapter 1

**Introduction**

 **OBJECTIVE**

The objective of our thesis is to use machine learning and a subfield of machine learning called **Deep Learning (DL)** to help us build algorithms that work like the human mind and are inspired by its structure. Information security professionals are also intrigued by such techniques, as they have provided promising results in defending against major cyber threats and attacks. One of the best-suited candidates for the implementation of DL is malware analysis.

**1.2 MOTIVATION**

Malware, which can be referred to by other names like malicious software or malicious code, can be described as “any code added, changed, or removed from a software system in order to intentionally cause harm or subvert the intended function of the system” [14]. Malware can be a piece of code attached to a legitimate program, an independent program, or a combination of both. The reasoning that leads to malware creation varies greatly, some are created to demonstrate some vulnerability or concept and do not cause direct harm to systems, while others are used to steal information or render systems useless. The wide variety and quantity of now-a-day systems and the amount of sensitive information they store provide numerous opportunities to illegally make a profit out of subverting legitimate systems. The amount of malware has shown an exponential increase since the early 2000s, from 1 million in 2006 to over 500 million by the end of 2016 [15]. According to a 2007 study by McAfee about 4% of the query results by major search engines lead to potentially dangerous websites. A more recent study by Norton in 2010 showed that at least 10% of top trending search terms returned malicious results.

To detect malicious software, anti-virus solutions rely on two main methods: signature-based, which uses a database of known malware to detect malware and heuristic-based, which make use of malicious patterns and sets of rules to detect both known and new malware [16]. To help in the task of detecting both new and old malware, the use of ML

And DL methodologies have shown promising results [17] when dealing with this task, but not without its pitfalls [18].

**1.3 BACKGROUND**

To the exponential growth of malware [19], the ability to distinguish between malware samples is crucial. In the biological world, different types of infections reckon distinctive disinfections; the same applies to the digital world. The infection method, behaviour and subsequently purpose of malware samples vary; hence being able to make a distinction facilitates prevention and disinfection by anti-virus solutions. There exists no ground truth when it comes to distinguishing malware, leading to subtle differences on the classification and naming of the same malware instance by different parties. To facilitate the reader’s understanding of the topic, the current work will focus on defining malware types based on their propagation and purpose.

With respect to the propagation method, malware can be split into three classes.

Virus: This type of malware inherits its name from resembling to biological viruses. A virus is usually composed by two main subroutines. The first subroutine is responsible for infecting other programs by attaching the virus code, while the second subroutine is the actual malware payload that contains the malware purpose (i.e. virus payload) [20]. Viruses are attached to programs and propagate when an infected program is run, either consciously (e.g. clicking on executables) or unconsciously (e.g. auto-run features), hence depending on other programs (as hosts) and user interaction. For viruses to infect new systems, an infected file must be carried between them by a user (e.g. USB pen, email attachment), reinforcing the user’s role in this propagation method.

Worm: The increasing number of network connected devices facilitates the application of worms to propagate malware. A worm shares the self-replicating ability of viruses, but discards user interaction and the need for a host. This is possible by making a worm an independent program that exploits networks to find vulnerable systems to infect with a copy of themselves . Similarly to viruses, worms can contain additional payload to perform malicious actions on infected systems (e.g. Blaster worm exploited vulnerability in RPC for propagation while its payload flooded a Microsoft domain1).

Trojan: While viruses and worms focus on self-propagation, Trojans focus on deceiving users into executing them, disregarding propagation. Trojans try to appeal users with some useful functionality as to allure into running the program [21]. By hoaxing users into running them, trojans bypass the need for a host (in the virus case) or an exploit (in the worm case) to perform malicious actions.

I explored the machine learning and deep learning applications in the fields of malware detection and did an extensive literature survey.

A good categorization of malware detection techniques was carried out by the authors in [1]. Apart from setting two main categories, anomaly-based and signature-based, the authors also split the detection algorithms by the method they use to collect information: static, dynamic and hybrid.

The anomaly-based technique has the advantage of detecting types of malware that were not present in the training set, while the static method of extracting features can be used in very large data sets.

Authors in [2] managed to obtain a 98.4% detection rate with 1.9% false positive. Even though one might consider that 1.9% is an acceptable false positive rate, it is still too big to be used in industry:

Network-based detection methods, including Honey pot-based approach [3-4], and based on Deep packet Inspection

[5]; Host-based detection methods, including check sum-based approach [6], signature-based approach [7-9], heuristic data mining approach [10].

The data mining method adopted many machine learning methods, which had an effective detection of unknown malicious code through learning the characteristics of malicious code

The normal code [11] reviewed a variety of feature-extraction methods and machine learning methods in a variety of malicious code detection applications, including naive

Bayes, decision trees, artificial neural networks, Support Vector Machine, etc., [12] Proposed a static system call sequences based on N-gram and two automatic feature-selection methods, and adopted K-nearest neighbor algorithm, SVM, decision tree as the classifier. The literature

[13] Presented a malicious code behavior feature extraction and detection method based on semantics to obtain the behavior of malicious code which has great antijamming capabilities.

Although the above methods have achieved certain results in the aspect of malicious code detection, there are still some problems. Such as, feature-extraction is not appropriate, the detection rate and the detection accuracy are not high, and the complexity of the algorithm is high.

**Chapter 2**

**PROJECT DESCRIPTION AND GOALS**

**2.1 PROJECT DESCRIPTION**

Our main focus is to implement the 3 algorithms. The first two algorithms are machine learning algorithms i.e. logistic regression and Random Forest algorithm.

The third algorithm is a deep learning algorithm, Neural Networks.

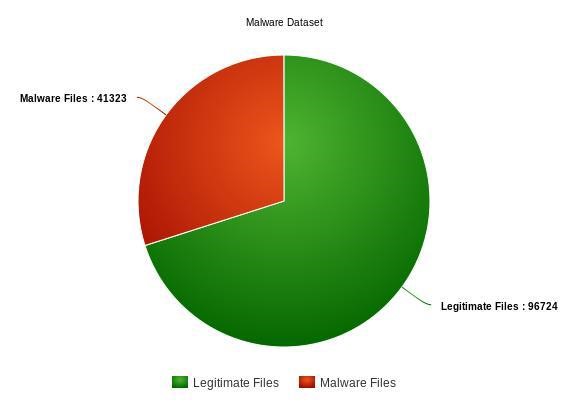
We will be comparing the results of all these algorithms and see which gives the best results for malware detection. We will be calculating the accuracy and f1 score for each algorithm.

**2.1.a DATA COLLECTION**

In this project, we are going to download a malware dataset delivered by a security blogger, Prateek Lalwani.

The malware dataset contains features extracted from the following:

* + - 41,323 Windows binaries (executables .exe and .dlls), as legitimate files.
    - 96,724 malware files downloaded from the Virus Share website.



*Figure 1: Dataset Distribution*

So, the dataset contains 138,048 lines, in total.

**2.1.b DATA PROCESSING**

In this step we will be cleaning and processing the dataset. As we know that there are many features in the dataset that our useless for feature selection, we will be removing the not so necessary columns from the dataset, removing duplicates and

removing null rows to increase efficiency of the algorithms.

We can remove the ‘md5’and the ‘name’ column as they do not contribute much to the machine learning and deep learning algorithms.

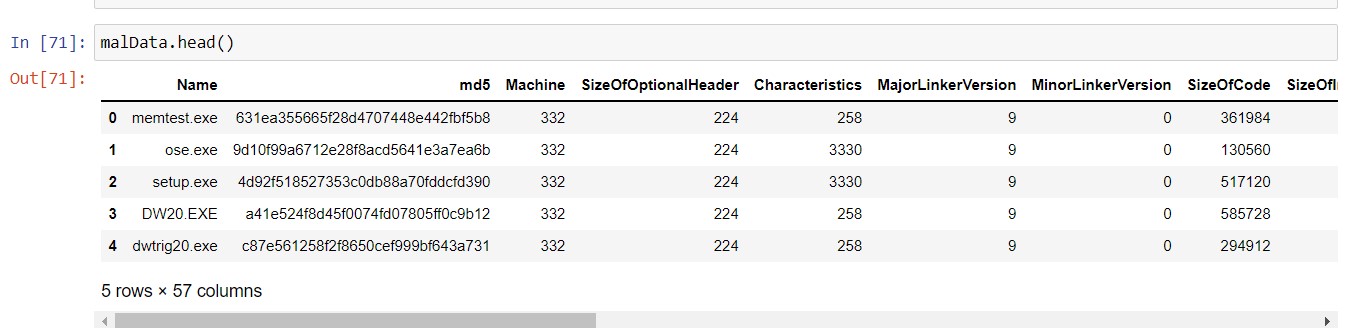


Figure 2: First Five rows of Dataset

**2.2 GOALS**

* The primary objective of the project is to successfully implement all the above mentioned algorithms and calculate the accuracy and f1 score along with required confusion matrix.
* Carry a detailed comparison between the algorithms by obtaining the accuracy results.

**Chapter 3**

**TECHNICAL SPECIFICATIONS**

**3.1. Hardware Specifications**

* + 1. 2.3GHz dual-core Intel Core i5
    2. Turbo Boost up to 3.6GHz
    3. 64MB of eDRAM
    4. 8GB of 2133MHz LPDDR3 onboard memory
    5. Intel Iris Plus Graphics 640

**3.2. Software Specifications**

* + 1. Python 3.7
    2. ScikitLearn
    3. Matplotlib
    4. Pandas and Numpy
    5. Jupytr Notebook
    6. Browser to run Jupytr Notebook (preferably Chrome)
    7. TensorFlow

**3.3. Modules Description**

1. Pandas

It is a vast open source library written in Python that enables you to perform data manipulation. It provides a simple way to create, control and change the data.

1. Numpy

It is the primary library of Python to provide scientific computing, which has a very powerful n-dimensional array object. It also provides tools for accommodating C, C++ etc. It is also helpful in linear algebra, capability of random numbers, etc.

1. Matplotlib

It is a library in Python specifically meant to plot data and its numerical maths extension NumPy. It gives us an object-oriented API to embed plots into various applications using general GUI toolkits.

1. Scikit-Learn

It is the fundamental library of Python specifically meant for Machine

Learning. It consists of various effective and efficient tools to perform Machine Learning and other statistical analysis like regression, classification, feature extraction,clustering, etc.

1. Seaborn

Seaborn is a Python data visualization library based on matplotlib. It provides a high- level interface for drawing attractive and informative statistical graphics..

1. Scipy

SciPy is a free and open-source Python library used for scientific computing and technical computing.

1. TensorFlow

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

Chapter 4

**DESIGN APPROACH AND DETAILS**

**4.1 ARCHITECTURE**

In this review we are implementing malware detection using Random forest, logistic regression and neural Network. To improve the estimators' accuracy scores, we are going to use the sklearn.feature\_selection module. This module is used in feature selection or dimensionality reduction in the dataset. To compute the features' importance, in our case, we are going to use tree-based feature selection.

**import sklearn**  **from sklearn.feature\_selection import SelectFromModel**  **from sklearn.ensemble import ExtraTreesClassifier**  **from sklearn.model\_selection import train\_test\_split**  **from sklearn import cross\_validation**

**Data = MalwareDataset.drop(['Name', 'md5', 'legitimate'], axis=1).values**

**Target = MalwareDataset['legitimate'].values**

**FeatSelect = sklearn.ensemble.ExtraTreesClassifier().fit(Data, Target)**

**Model = SelectFromModel(FeatSelect, prefit=True)**

**Data\_new = Model.transform(Data)**

**print (Data.shape)**  **print (Data\_new.shape)**

The above commands and code will show us the selected features that will be used for training,

After feature extraction we will split the dataset and 20% of the data will be used for testing and rest of the data will be used for training purposes.

Further in the project the following models will be explored in great detail:

* Logistic regression
* Random Forests
* Neural Networks (Deep learning)

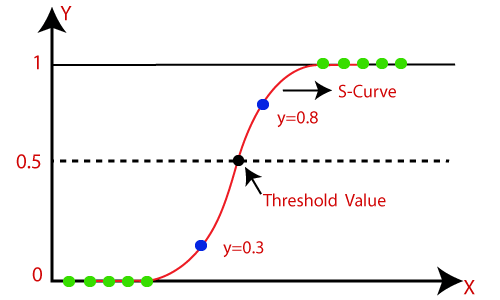
**4.2 LOGISTIC REGRESSION**

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes.

In simple words, the dependent variable is binary in nature having data coded as either 1 (stands for success/yes) or 0 (stands for failure/no).

Mathematically, a logistic regression model predicts P(Y=1) as a function of X. It is one of the simplest ML algorithms that can be used for various classification problems such as spam detection, Diabetes prediction, cancer detection etc.

Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. The below image is showing the logistic function:



# Figure 3: Logistic Regression

Advantages of Logistic Regression are:

1. Logistic regression is easier to implement, interpret, and very efficient to train.
2. It can easily extend to multiple classes (multinomial regression) and a natural probabilistic view of class predictions.
3. It is very fast at classifying unknown records.
4. It can interpret model coefficients as indicators of feature importance.
5. Logistic regression is less inclined to over-fitting but it can overfit in high dimensional datasets. One may consider Regularization (L1 and L2) techniques to avoid over-fitting in these scenarios.

Disadvantages of Logistic Regression are:

1. If the number of observations is lesser than the number of features, Logistic

Regression should not be used, otherwise, it may lead to overfitting.

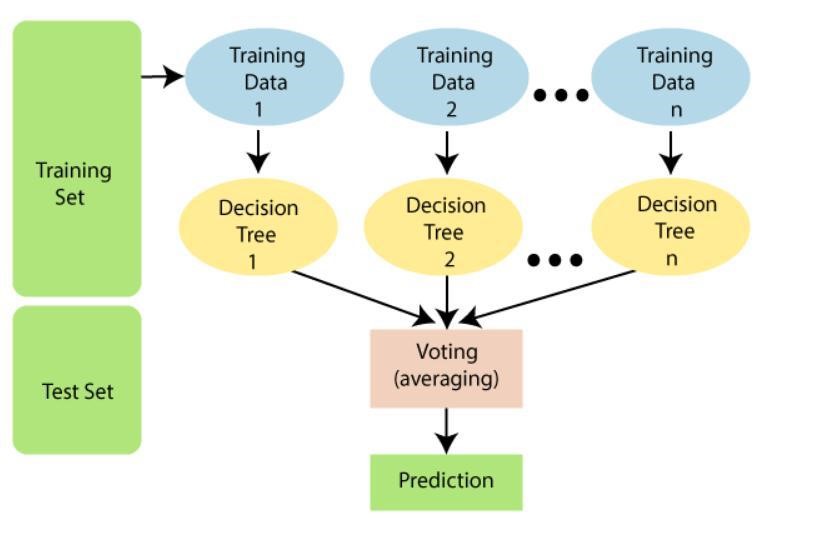
1. The major limitation of Logistic Regression is the assumption of linearity between the dependent variable and the independent variables.
2. Non-linear problems can’t be solved with logistic regression because it has a linear decision surface. Linearly separable data is rarely found in real-world scenarios.

**4.3 RANDOM FOREST**

It is a Supervised Learning algorithm which employs ensemble learning methods primarily used for classification and regression.

Decision trees are very sensitive to that particular data on which training is done. If the data on which it is trained is changed, then the resulting decision tree can be extremely different and thus the predictions may come out to be very different.

Decision trees are way too computationally expensive to train as they require a lot of computing power, carry a big risk of over fitting on the data, and mostly try to find the local optima as they cannot go back after they have made their respective split. To overcome these disadvantages, we switch to Random Forest which demonstrates the power of mixing many decision trees into one powerful model.



# Figure 4: Random Forest algorithm with different decision trees

Random forest is a bootstrap aggregating (also called bagging) technique and not a boosting technique. The trees in the random forest mostly run in parallel. While building the trees, there is little to no interaction between the trees. It works by making a large variety of decision trees during the training period and returning that class as the output which is the mode of the classes (in the case of classification) or the mean prediction (in the case of regression) of the individual trees.

A random forest is a meta-estimating algorithm (i.e. it mixes the output of multiple predictions) which includes many decision trees, with some needful changes:

The number of variables that can be split on at every node is limited to a few percentage of the total (also called the hyperparameter). This makes sure that the ensemble model does not depend too much on any single feature, and makes fair usage of all the important predictive features.

Each tree extracts a sample at random from the original data while making its splits, adding an additional element of randomness that reduces the chances of overfitting.

The above modifications help prevent the trees from being too highly correlated. Some of the main features and advantages of the Random Forest algorithm are:

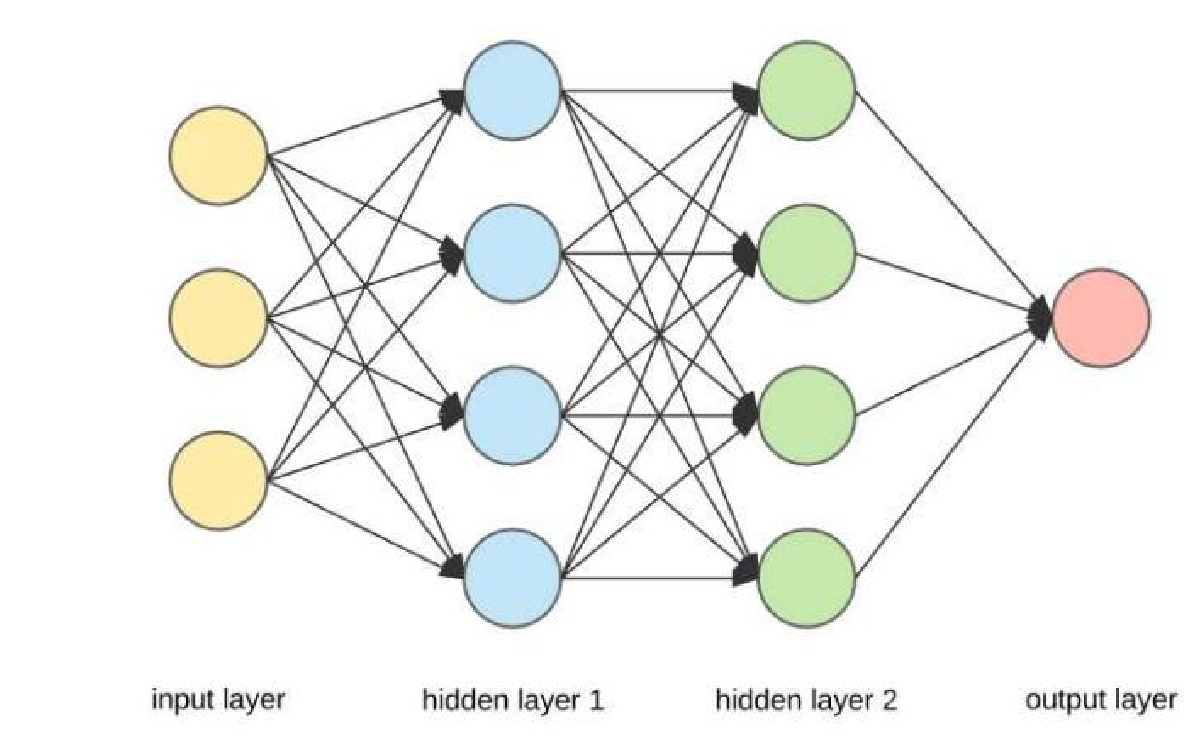
1. It is probably the most accurate learning algorithm out of those currently available. For a lot of data sets, it produces a very accurate classifier.
2. It can run easily and elegantly on large databases.
3. It has the power to handle hundreds of input variables without deleting any variable.
4. It gives a decent estimate of the variables that are important for the classification.
5. It generates an internal unbiased estimate of the generalization error as the forest building progresses.
6. It has an efficient method to estimate any missing data and thus maintaining accuracy when a large part of the data is missing.

Some of the disadvantages of Random Forest are:

1. Random forest algorithm has been found to overfit for the datasets consisting of noisy classification or regression tasks.
2. For datasets having categorical variables with different numbers of values, random forests favor those attributes which have greater value. Therefore, the variable importance scores obtained from random forests cannot be relied upon for such data.

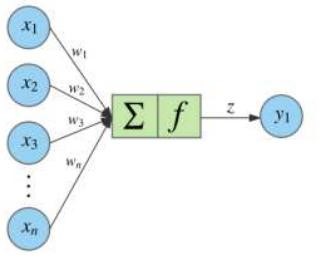
**4.4 NEURAL NETWORK**

Neural Networks (NN) are neural nets having completely connected multiple layers. They include an input layer, several hidden layers, and output layer. Each node in a layer is connected to every other node in the proceeding layer. We can make the network denser by raising the number of hidden layers in the network.



# Figure 5: Neural Network with all Layers

A deeper look at the output node would show that it looks actually like this:



# Figure 6: Nodal view of neural Network

A given node takes in the sum of its inputs in a weighted manner, and passes that weighted sum via a non-linear activation function. This processed part is the output of the node, which then becomes the input of further nodes of the proceeding layer. The flow of signal is from left to right, and the last and final output is determined by doing this procedure for every node. To train this neural network will mean to learn the weights which accompany all the edges. The weighted sum of the inputs it received is passed via a non-linear activation function. It can be presented as a vector dot product, where n denotes the number of inputs for that particular node. Once the training is done, we only execute the forward pass to predict. But before that, we need to train our model to learn the weights, and that procedure works like this:

1. We initialize randomly the weights of every node. There are many good

initialization methods which we can use.

1. For every training set, we do a forward pass using the existing weights, and

determine the output of each and every node from left to right. The final and last

output is the value of the ending node.

1. Compare the resulting final output with the actual value of the target of the

training data, and measure the resulting error using what is called as a loss

function.

### 4. Do a backwards pass in the reverse direction and propagate the error to every

single node by implementing backpropagation. Find each weight’s contribution to

the resulting error, and adjust the then weights accordingly by the help of

gradient descent. Take the error gradients back to the starting from the final layer.

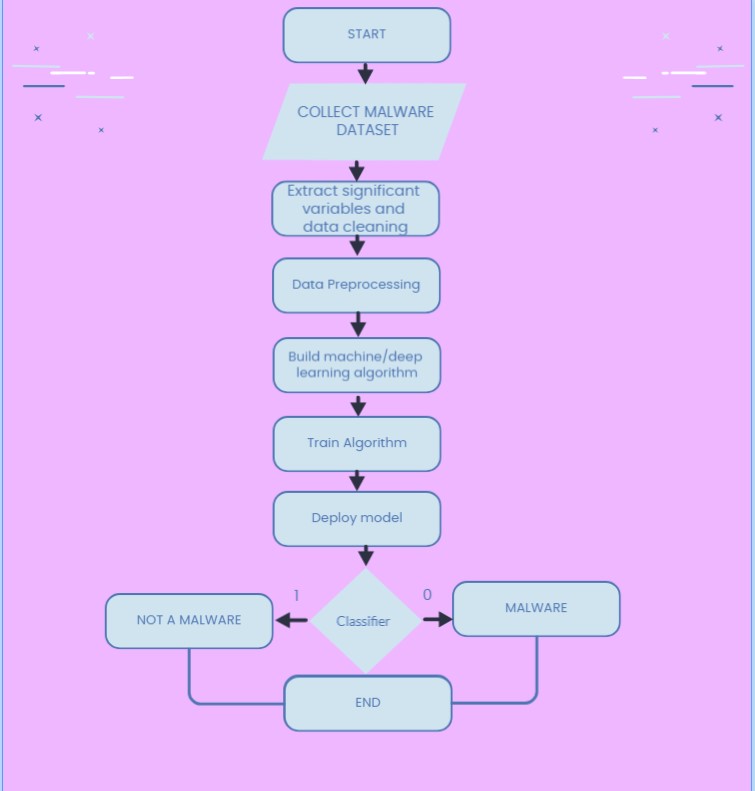
Some of the advantages of Artificial Neural Networks ( ANN) are :

1. Saving information on the entire network: Information like in traditional programming is saved on the whole network, not on a regular database. The vanishing of some pieces of information in one domain does not stop the network from working properly.
2. Able to function with partial knowledge: After the ANN is trained; the data may produce the output even with partial information. The loss in performance here majorly depends on the weightage of the missing information.
3. Fault Tolerant: Degradation of one or more values of ANN doesn’t stop it from generating output. This advantage makes the networks more robust and fault tolerant.
4. Has a distributed memory: In order for ANN to learn efficiently, it is important to find the examples and to teach the network in accordance with the expected output by presenting these examples to the network. The network's successful performance is directly proportional to the instances that are selected, and if the event can’t be presented to the network in the major aspects, the network can produce wrong output
5. Incremental corruption: A network slows down with time and thus undergoes a relative degradation. However, the network problem does not get corroded immediately.
6. Ability to employ machine learning: ANNs learn through events and make the respective decisions by taking on similar events.
7. Parallel processing abilities: Artificial neural networks have the unique advantage of performing more than one task at the same time.

Some of the disadvantages of Artificial Neural Networks (ANN) are :

* + 1. Hardware dependence: Artificial neural networks need parallel processing power, according to their structure. Because of this, the realization of the equipment is dependent on the network.
    2. Unexplained behavior of the ANN: This is the most crushing problem of ANN. When ANN produces a wrong solution, it does not give any hint as to why and how the error occurred. This decreases the trust in the ANN.
    3. Determining the correct network structure: There is no single rule to fix the structure of artificial neural networks. The required network structure is constructed through experience or/and trial and error.

**4.5 PROJCT FLOW DIAGRAM**



# Figure 7: A project flow diagram

Activity diagram is defined as a Unified Modelling Language diagram that concentrates on the development and flow of the system's behaviour instead of its implementation. It is also sometimes referred to as an object-oriented flowchart. Activity diagrams comprise of activities that are made up of actions which apply to behavioural modelling technology.

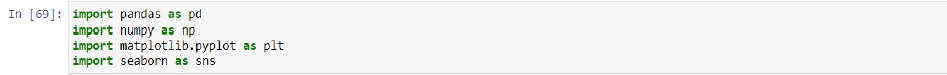
The Activity Diagram for this project is given above.

## Chapter 5

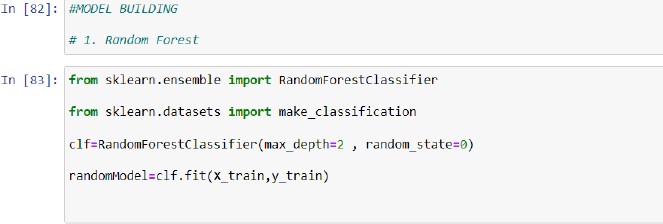
**PROJECT DEMONSTRATION**

**5.1 IMPORTING MODULES AND DATA CLEANING**

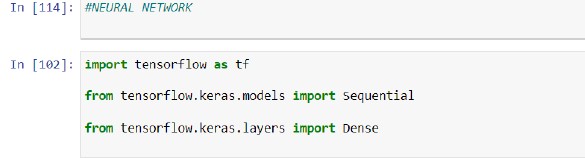
In this step we will be cleaning the dataset by removing duplicates and removing those columns which have little to no contribution in the below mentioned algorithms .We will also be dropping the ‘legitimate ‘column and storing it in a different variable for testing and training purposes.. After that we will be importing the modules required for building our machine learning and deep learning algorithms.



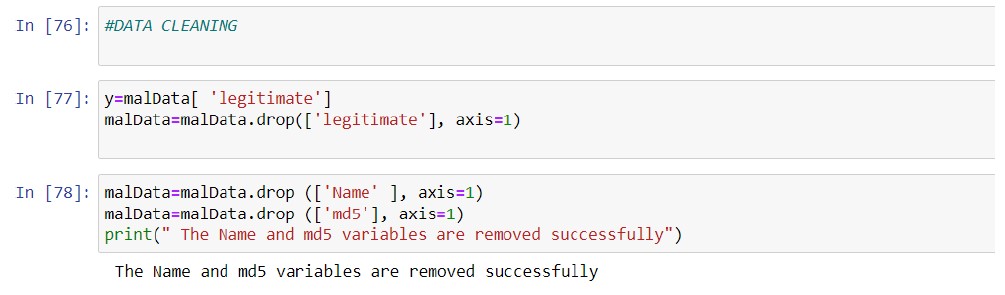
# Figure 8:Importing Modules



# Figure 9:Importing Modules for Random Forest



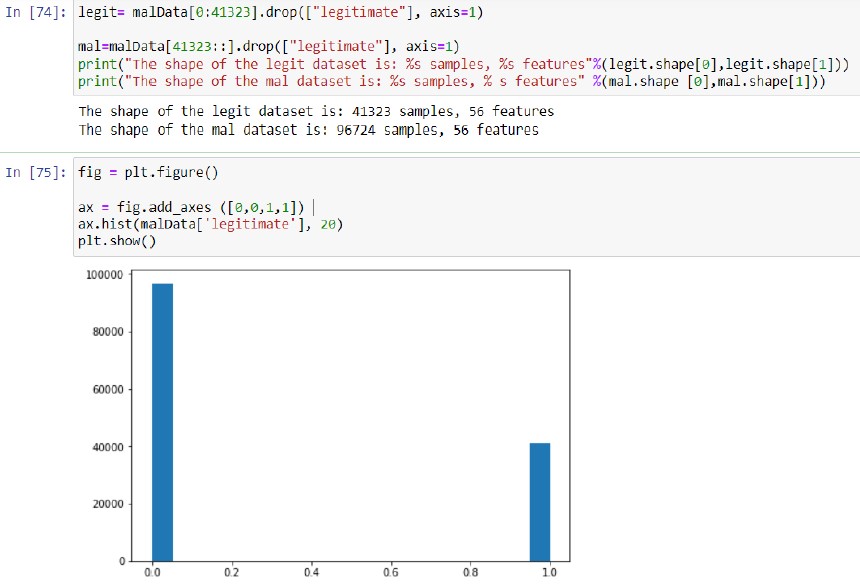
# Figure 10: Importing Modules for Neural Network



# Figure 11: Cleaning Data

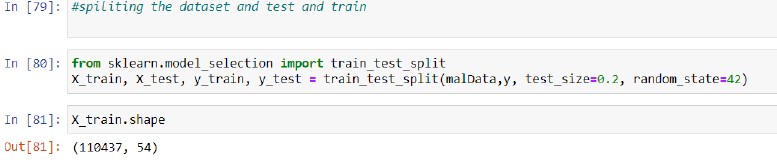
**5.2 DATA PREPROCESSING**

In this step we will be storing the legit and malware data into two different variables. I have also plotted a graph showing the disparity in the dataset. We can clearly see that malware files are greater in number in comparison to the legit files.



# Figure 12: Data PreProcessing

Now we will be splitting the data into four parts for testing and training both the legit and malware files.



# Figure 13: Data Splitting

**5.3 LOGISTIC REGRESSION IMPLENTATION**



First we will be importing the logistic regression module

**5.4 RANDOM FOREST IMPLEMENTATION**

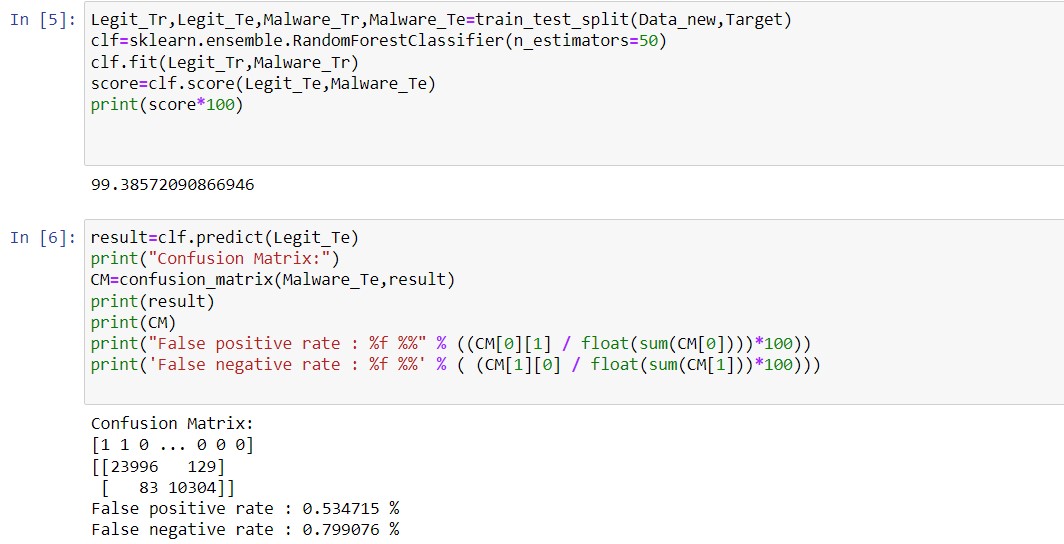
First we will import the modules



# Figure 15: Importing Modules

Now we will be using ExtraTreesClassifier for feature extraction that will give us the most important attributes that we will be using in our random forest model

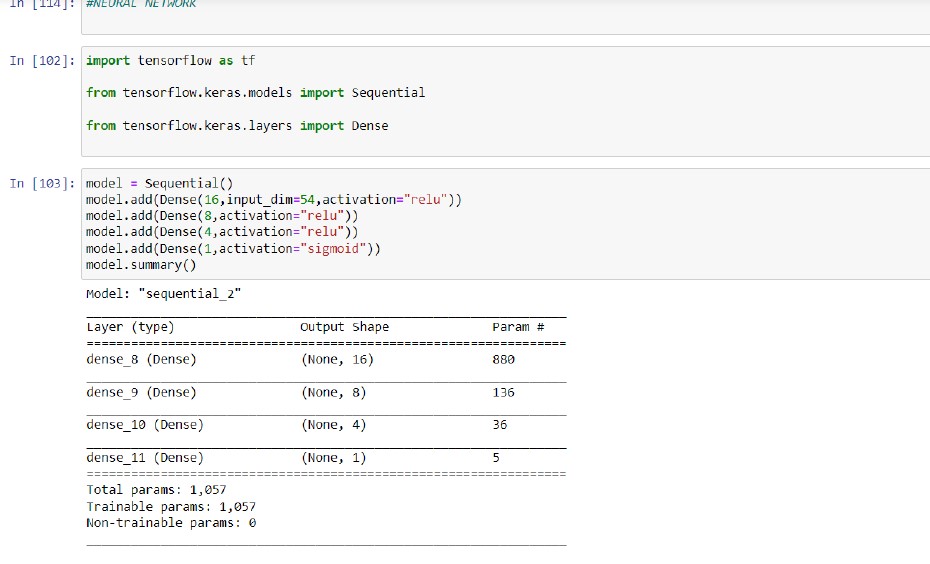




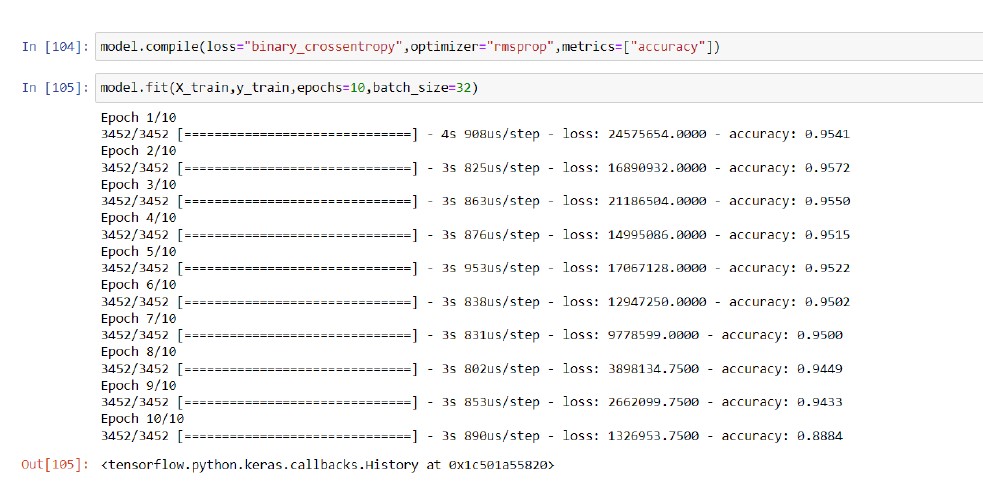
# Figure 17: Implementation of Random Forest

**5.5 NEURAL NETWORK IMPLEMENTATION**

I have taken RELU activation function in 3 layers and in the last layer I have taken the sigmoid activation function because the output had to be either 0 or 1.



# Figure 18: Creating a model for neural Network

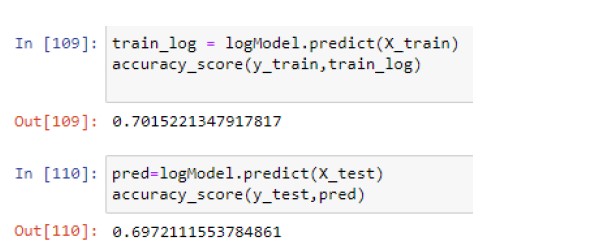


# Figure 19: Epoch Training for neural Network

**Chapter 6**

**RESULTS**

6.1 LOGISTIC REGRESSION

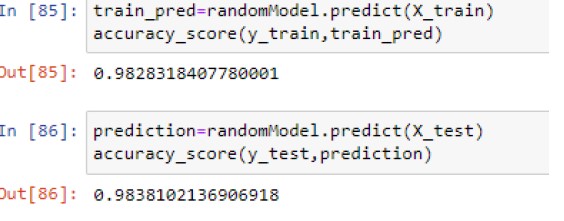


# Figure 20: Accuracy of Logistic Regression



# Figure 21: Confusion Matrix of Logistic Regression

**6.2 RANDOM FOREST**

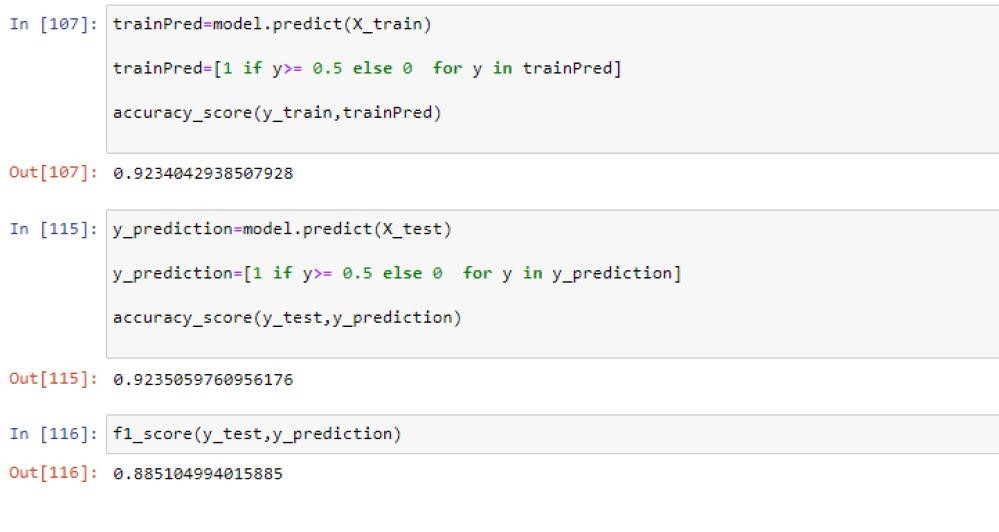


# Figure 22: Accuracy of Random Forest



# Figure 23: Confusion Matrix of Random Forest

**6.3 NEURAL NETWORK**



*Figure 24: Accuracy and f1score of Neural Network*

Chapter 7

**CONCLUSION AND FUTURE WORK**

The solution for the problem statement has been tested via 3 types of algorithms, traditional Machine Learning algorithms like Random Forest, logistic regression and a Deep Learning algorithm like ANN.

By observing the results of all the algorithms that were carried out in the dataset, we can clearly see that random forest classifier was by far the best in accuracy of 98% and f1 score of 0.95. Whereas logistic regression had the lowest accuracy of 70.1% and f1 score of 0.69. Neural network fared well with an accuracy of 92% and f1 score of 0.88. We also have to keep in mind that the dataset was highly disperse as the number of malware files were 96,724 on the other hand legit files were only 41,323. Due to this we can see that logistic regression approach gave us a very low accuracy as the database was imbalanced while random forest gave us a very good result due to the fact that random forest classifier constructs several trees which compensates for the fact that the dataset was imbalanced. The results of neural network can be improved by changing the epochs and trying out new activation functions.

Future Works:

1. Testing with different numbers of nodes and hidden layers and finding a way to automate that task.
2. Diverging from a supervised learning methodology to include other types of learning would also bring benefits and possibly achieve a fully automated pipeline to detect malware, with minimal user interaction.
3. Handle ‘md5’ attributes which we neglected for simplicity purposes.

**REFERENCES**

Idika, N., Mathur, A.P(February 2007).: A survey on malware detection techniques. PhD thesis. Purdue University

Siddiqui(2008), M.A.: Data mining methods for malware detection

C. L. Tsai, C. C. Tseng and C. C. Han, Editors (2009), “Intrusive behavior analysis based on honey pot tracking and ant algorithm analysis”, 43rd

Annual 2009 International Carnahan Conference

W. Wang and I. Murynets (2013), J. Security and Communication Networks, vol. 6, no. 1.

P. C. Lin, Y. D. Lin, Y. C. Laiand and T. H. Lee(2008), J. Computer Practices, vol. 41, no. 4.

Y. Sawaya, A. Kubota and Y. Miyake, Editors (2011) , “Detection of attackers in services using anomalous host behavior based on traffic flow statistics”, IEEE/IPSJ 11th International Symposium

M. Milenkovic, A. Milenkovic and E. Jovanov, J. ACM SIGARCH(2005) Computer Architecture News, vol. 33, no. 1.

M. Christodorescu, S. Jha, S. A. Seshia, D. Song and R. E. Bryant, Editors,

(2005)Proceedings of the 2005 IEEE Symposium Security and Privacy

M. Christodorescu and S. Jha(2006), “Static analysis of executables to detect malicious patterns”, Wisconsin: University of Wisconsin.

D. G. Kong, X. B. Tan, H. S. Xi, T. Gong and J. M. Shuai(2011), J. Journal of Software, vol. 22, no. 3

A. Shabtai, R. Moskovitch, Y. Elovici and C. Glezer(2009), J. Information Security Technical Report, vol. 14, no. 1.

Y. X. Ding, X. B. Yuan, D. Zhou, L. Dong and Z. C(2011). An, J.

Computers &Security, vol. 30, no. 6.

R. Wang, D. G. Feng, Y. Yang and P. R. Su (2012), J. Journal of Software, vol. 23, no. 2.

G. McGraw and G. Morrisett(2000). Attacking malicious code: A report to the infosec research council. IEEE software

McAfee Labs Threats Report March(2018).Rp-quarterly-threats p24.

D. Gryaznov(1999). Scanners of the year 2000: Heuristics. In Proceedings of the 5th International Virus Bulletin, volume 113.

N. Srndic and P. Laskov(1999). Detection of malicious pdf files based on hierarchical document structure. In Proceedings of the 20th Annual Network & Distributed System Security Symposium.

A. Deo, S. K. Dash, G. Suarez-Tangil, V. Vovk, and L. Cavallaro(2016). Prescience: Probabilistic guidance on the retraining conundrum for malware detection. In Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security, p 71–82.

AV-TEST Security Report 2016-2017(2017). AV-TEST\_Security\_Report p

34.

T. M. Chen and J.-M. Robert(2004). The evolution of viruses and worms.

Statistical methods in computer security, 1.

P. Szor(2005). The art of computer virus research and defense. Pearson Education.

**APPENDIX A**

|  |  |
| --- | --- |
| **Tasks** | **Contributions** |
| Literature Survey | Himanshu |
| Collecting Dataset | Arnav Malhotra |
| Pre Processing | Aakash Kumar |
| Feature Extraction | Himanshu |
| Logistic Regression | Aman Chaudhary |
| Random Forest | Arnav Malhotra |
| ANN | Aman Chaudhary |
| Comparison, Compiling and Documentation of Results | Aakash Kumar |

*Table 1. Contribution by Members*