Detection of malware in a file using Machine Learning

MSc Academic
Cybersecurity

Saishankar Murali
Student ID: - x18174990

School of Computing
National College of Ireland

Supervisor: - Prof. Imran Khan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Saishankar Murali

Student ID: X18174990

Programme: MSc Cybersecurity **Year:** 2019 -2020

Module: MSc Academic Internship

Supervisor: Mr.Imran Khan

Submission Due Date: August 17th 2020

Project Title: Detection of Malware in a file using machine Learning with KNN model

Word Count: 6878 words **Page Count** 23 pages

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.
ALL 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.
I agree to an electronic copy of my thesis being made publicly available on NORMA the National College of Ireland's Institutional Repository for consultation.

Signature: Saishanar Murali

Date: 17.08.2020

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):	

Table of Contents

Table of Contents

Detection of malware in a file using Machine Learning with KNN Model

Saishankar Murali
x18174990

Abstract: -

In today's world the technology is increasing very rapidly and so is the increasing the development of malware and malicious activity through which the cybercriminals are gaining a lot on sensitive information and credentials. The innovations of the latest technology motivate cybercriminals to create malicious code and also to perform them through which they steal data and perform abnormal activities in the technologies. This is the reason malware detection is very important so that we can detect the malware at the early stage and prevent the devices from getting attacked. There are many detection methods established but to enhance them more further machine learning technique can be used because it is accurate and efficient. In this paper I will be using machine learning techniques, K-means clustering algorithm to detect the malware with the help of dataset which will be trained, Kmeans clustering algorithm will be used because it is accurate and gives the output as expected because of the mathematical operation it consists. In this, we will be evaluating the performance based on the algorithm, Detection, and the dataset that will be used, and also confusion matrix will be performed to get the false positive and negative results.

Keywords: - Machine Learning, Malware, Detection, Algorithm, Dataset, Private Information.

1.Introduction

Technology and their latest upgrades and updates are increasing day by day and have a very great impact on our day to day life. Each and everyone is surrounded by technology which is making their life much easier. But this is where cybercriminals have many opportunities to sabotage the new technology by spreading malware in various forms such as through emails, links, documents, etc. This happens because development teams of any IT industry mainly focus on the interface and design of the software instead of securing their network or any method that will prevent from entering of malicious files. But some of the companies are investing or spending billions and billions of money to secure their software to the security organization. Also, researchers and development of cybersecurity are working and doing excellent work on establishing new security approaches to prevent cyber attacks.

To explain the term malware, it is nothing but a piece of software that is developed by a cyber attacker in intent to damage a technology device by stealing information, credentials, etc. Malware is of different types such as Viruses, Worms, Spywares, Botnet, Ransomware, Trojans, etc. Malware is created by attackers and they only develop it to make money or by any means to gain profit, they also gain money and profit by selling it to the dark web whose bids are the highest. Whereas these are the disadvantage of creating malware but the advantage of the malware is that the companies also buy malware to test the security of the software that they developed. Malware on the other hand creates great damage to the reputation of the company and also lots of loss to the company by revealing sensitive information or stealing it. According to McAfee, it is said that the new type of malware is detected every four seconds. It is said that even the antivirus software is not able to detect

the types of malware for eg. according to research it seen that 69,477,489 unique malicious viruses have gone undetected. Each time the malware doesn't need to be created in the same way or method, it can change its form being stealth and can go undetected.

To mitigate these kinds of problems a better approach for this would be the use of machine learning techniques and their different algorithms because of which the detection rate will much improve to detect malware. The software that will be created will run machine instructions that are created by python and give us accurate results. In this we will be downloading a dataset that will be trained to detect malware and the use of the K-means clustering algorithm will be very helpful to accurately cluster the malware accordingly.

Research Question: - How can we detect malware by using machine learning with the help of?

The research paper is divided into the following sections: Section 2 which literature review focuses on the other research paper of detection of malware which was proposed by other researchers. Section 3 will consist of the research methodology and section 4 will be the design specifications of the research. Section 5 will be the main part which is the implementation where how the model is made will be discussed in detail.

2. Literature review:

In this section we will be discussing the implementation of malware detection which are proposed by different authors, we will be seeing how they have used different types of detection algorithms and how they have tested their model, also we will be comparing them further.

In this paper proposed by Santosh Joshi, Himanshu Upadhyay, Leonel Lagos, Naga Suryamitra Akkipeddi, and Valerie they have used machine learning classifiers to detect the behavior of the malware. They have used the Random Forest algorithm which is the traditional way to detect and analyze the malware that causes cyber threats. The proposal tells that a given sample is classified as a benign or malware with high accuracy and low low computational overhead. The parameters to perform this used by the author are Linux environment, the data is collected is extracted by using LibVMI which is specially made for Virtual Machine Introspection. The authors have used virtualization method because it provides better approaches to create a virtual machine and then creates or run the malware to analyze the behavior of the malware, this is done because if the malware is executed in virtualize environment then it cant affect the host machine. [1] Xen hypervisor-based virtualization platform is used to host the system, the data is captured by virtual memory introspection using the library LibVMI and then transfer everything into the database. The dataset that was used to build the model consisted of a mixture of malicious malware and benign data which were extracted from the list data of the Linux VM. [1] Dataset had 100,000 benign processes and 109,998 malware processes. For training the data they split the data into two parts such

as 70% of the data was used for the training purpose and 30% was used for the testing process, along with this they used the confusion matrix for the evaluation and plotted the accuracy. The author has stored the data in a database server in the virtualization platform and then executed the Random Forest algorithm by using the machine learning server. The model that has been created will be stored in the database for prediction. To achieve the accuracy for the model the author does c]some minor changes into the random forest algorithm such as changing the number of trees, number of variables that are tried at each split. According to the testing and training of data, the author can achieve accuracy of 90.9%[1]. In last the authors conclude that machine learning techniques are very promising towards detecting the malware in this technological world, also various models are proposed but without testing and achieving accuracy no one cant decides which one to employ.

In contrast to the above-proposed method, Ihab Shhadat, Bara Bataineh, Amena Hayajneh, and Ziad A. AL-Sharif have proposed the detection of unknown malware using machine learning to advance the detection and classification. The author has stated that the purpose of this research to detect the unknown malware using machine learning techniques, the author has applied different types of the algorithm on the dataset to check which gives better accuracy in binary and multi-classifiers. The author has mentioned that they have achieved by using the Random Forest classifier for the feature selection and cross-validating the data. The data that was collected had 1156 files in total out of which 984 were malicious files and 172 benign files, and the formats in which data were collected were of different forms such as .exe, .pdf and .docx[2]. The author has considered various types of main malware in their study such as Drider, Locky, Zeus, TeslaCrypt, etc.[2]. They have estimated the performance based on Accuracy, precision, Recall, and the F1-score which is calculated with the help of the confusion matrix. In this research paper, feature extraction has been conducted using the heuristic strategy which is a dynamic analysis in which malicious files are executed in a virtual environment and then information is gathered based on the malware characteristics. But in this, the author has acquired only the needed information which will give a better accuracy rate and better performance and rejecting the rest of the information which are not needed because they can give problems in the accuracy rate and performance. Random Forest algorithm is used because they often decide the important features accurately, it is a treebased technique that ranks the impurity of the node. Cross-validation is done because evaluation is very important in machine learning, K-folds-cross-validation divides the initial sets into k subsets into a similar size then performs the training[2]. According to the research paper the important features were specified with the help of Random forest and the crossvalidation was done by using K-fold-cross-validation The author have applied different types of model which are provided by the sklearn. [2] The highest accuracy was achieved by Decision tree(98%), Random Forest(97.8%), Hard Voting(97%), KNN(96.1%), SVM(96.1%), LR(95%). The lowest accuracy was achieved by Bernoulli NB which is 91%. The author concludes that the performance that is studied of machine learning techniques in which the decision tree has achieved 98.2% accuracy which out-performs the rest of the algorithm.

In this paper, the authors Deepak Gupta and Rinkle Rani have proposed a method which is improving malware detection using big data and ensemble learning. The authors have

proposed two methods to improve the detection of malware, the first one is the weighted voting strategy of ensemble learning and the second one is choosing an optimal set of base classifiers to stack. The two methods that are mentioned above are based on ensemble learning and big data which improves malware detection at a very large scale. It is mentioned that the malware and the benign files are collected from the public repositories and clean windows machine, to perform static and dynamic analysis author has set up an automated environment to generate reports which are used as a raw dataset. These data are stored in a distributed storage system and processed using Apache spark to extract the features. The dataset for the proposed method consists of a 100,200 malware file and 98,150 benign files[3]. The malware files were collected from sources like VXHeven, VirusShare, etc. and the benign files were collected from the clean Windows XP, 7, 10, and installation directory. The author has also said that the dataset is nearly balanced because the difference between the malicious file and benign files are less so they are almost equal because of which better accuracy and detection can be achieved. The use of an automated environment has been used to perform the static and dynamic analysis, talking of the setup the host machine is Ubuntu as OS and Windows as a guest machine which is created in Oracle VM VirtualBox. The binary files are executed in the virtual environment and generate a report of full analysis in JSON format. The feature extraction for detection has been deciding based on the reports of dynamic and static analysis of which the report was generated and the feature extraction is File metadata, Filesize, Packer detection, Sections Information, Dynamically linked libraries, Dropped files, Windows API calls, Mutex operations, File activities, Registry Activities, Network activities and Process activities[3]. These are the features based on which further detection will take place. The algorithms on which the dataset has been applied in this paper are Naïve Bayes, K-nearest neighbor, Decision table, SVM, and Random Forest. After applying the dataset in every algorithm the author says that Random Forest has given the highest accuracy which is 98.1% followed by the Decision table algorithm which is 94.9%. The author concludes that the proposed system that is the weighted voting technique gives the highest accuracy of 99.5%, hence that proposed system enhances the detection of malware.

In contrast to the above paper, In this paper, the author's Priyank Singhal and Natasha Raul have proposed a method which is a Malware detection module using machine learning algorithms to assist centralized security in enterprise networks. In this paper they have proposed a new and much complex antivirus engine that can scan harmful files, this process is done by extracting the API calls made by different types of normal and malicious executables. Machine learning will be used to enhance classify in a better way and also rank the files based on their security risk. This system is very heavy because of the processor but it is very effective and can be used in enterprise networks to detect threats. In this research paper, the solution of detecting the virus works on a firewall level of the enterprise network. For detection the authors have extracted numerous infected and normal PE header executables by using IAT, then they store the extracted data into a data mine. The information gain is derived for each function[4]. For the further implementation, Random forest is used, it works as a classifier to detect the malware, Random forest's output prediction will be the most frequent class output of the individual trees. The author has said that to check whether the proposed model can provide results or not they have extracted over 5000 executable files

which consist of both normal and infected, by using the information gain algorithm they chose only the 80% of functions that are more likely to be harmful. In the paper the results show that the accuracy achieved using different algorithms which are Decision tree has 90%, Naïve Bayes has 95%, Random Forest has 97% and the proposed method has got 98%. The author has concluded that the model that has been proposed can detect malware based on advanced data mining and machine learning techniques. Whereas this model cannot be used for home users or others but it can be used only on an enterprise level.

In this research paper, the author's J. Zico Kolter and Marcus A. Maloof have proposed a method which is to detect and classify malicious executables in the wild. The author has said that 1,971 benign files and 1,651 malicious executables and each of them was encoded as a training example using n-gram as features. After selecting the features the authors have said that they applied various algorithms for evaluations such as Naïve Bayes, Decision trees, Support vector machines, and boosting[4]. As stated in the paper boosted decision trees have outperformed other algorithms with a ROC curve of 0.996. The authors have observed that their methodology can scale to a larger number of executables and also how their method classifies the executables based on their payload. As mentioned above the data consisted of 1,971 benign files and 1,651 malicious files which were collected in the window PE format, and also the benign files were collected from the folders of windows XP and Windows 2000 machines. The hex dump was used to convert the executables into hexadecimal codes in ASCII format so that the machine can understand.[5] Then the respected n-grams were created by combining each four-byte sequence into a single term, for example, if a sequence is ff 00 ab 3e 12 b3 then the corresponding n-gram sequence will be ff00ab3e, 00ab3e12 and ab3e12b3. The experimental results say that the rate of false-positive rate is 0.05 and the accuracy of a boosted decision tree is 98% of malicious files out which 6 went missing from the 291 malicious files, but the author says that for some 6 maybe a major issue if one is ready to accept a false positive rate of 0.1 then it can detect with perfect accuracy. The author concludes that after all the evaluation boosted J48 outperforms other algorithms and is the best detector because of the ROC curve of 0.996.

In this paper, the author's Yanfang Ye, Digging Wang, and Dongyi Ye have proposed a method which is called an Intelligent Malware detection system. The author has said that by analyzing Windows API execution processes which are called PE header, they have implemented an intelligent malware detection system using objective -oriented association mining based classification. An experimental study with a large section of PE files that were obtained from the anti-virus laboratory of King-soft corporation is used to compare various malware detection. The author has said the statement that the IMDS system outperforms popular antivirus software such as Norton, McAfee and virus scan, etc and also other algorithms that are normally used. The results show that the IMDS has got the highest accuracy which 93.07% and outperforms other algorithms such as Naïve Bayes, SVM, and J4.8. The author has concluded that they have implemented the model successfully with a large collection of 12212 benign samples and 17366 malicious files.

In relation to the other papers, in this paper, the author's C.P. Patidar and Harshita Khandelwal have proposed a method to detect a Zero-day attack using machine learning techniques. The author has divided this into four phases namely the Malware datasets, Analyzing MDS, correlation algorithm, Detection methods of malware[7]. Firstly the sample of malware is collected which are called datasets, then the use of correlation algorithm is used to correlate the relationship between the malware and will be able to predict the future. The final phase is to apply the detection model which will detect the malware and gives the appropriate output.

3. Research Methodology

To implement the model and also to evaluate the model we have used the CRISP-DM methodology. It is a cross-industry process for data mining and it provides a very good structured approach to plan a data mining project. The image below illustrates the process of the CRISP-DM methodology, further the steps related to the CRISP-DM are explained below.

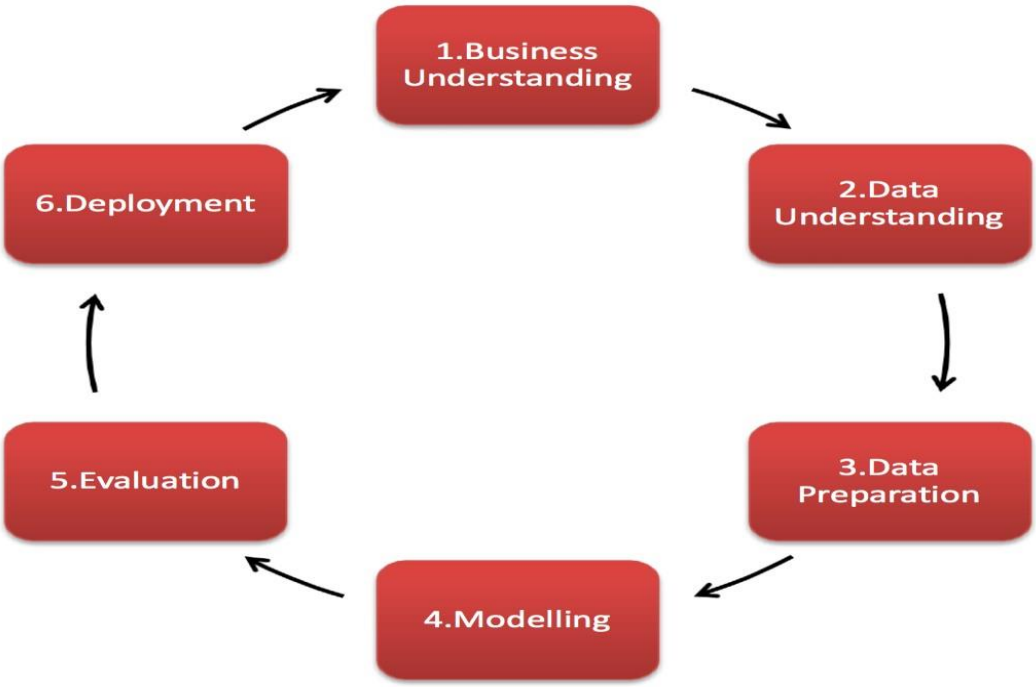


Figure1: Process of implementation

3.1 Business Requirement:

Malware attacks in this technological world are increasing day by day and it is causing lots of damage to the organization and also to the reputation. Anti-viruses like Norton, McAfee, Quick heal just won't be enough because they use signature-based detection out of which many of the signatures go missing and the damage is still caused. To enhance the detection we have to use machine learning techniques

Many research papers have been proposed to detect the malware based on their various features and the characteristics, but most have them have used less data which gives the accuracy as high as possible but what if the data is on a large scale and doesn't give the same output?. So this research is to implement a solution to the problems of malware affecting the organizations and companies. Using machine learning can be a better solution because we can detect the malware by analyzing the behavior of it, also we can detect the malware even if the data is on a large scale.

3.2 Data Acquisition:

According to research conducted there are many open-source datasets available on different websites such as Kaggle, VirusTotal, VirusShare, etc, But when compared to each other the datasets that are available on Kaggle are much better and the data are balanced. In Kaggle there is a dataset named Brazilian malware and goodware dataset which will be a good start. This dataset consists of 35,694 malicious files and 21,164 benign files.

The Dataset was downloaded from the Kaggle website, the website had two files in separate which consisted of malware files and the other one was the good ware, both files were in .csv format so we could load the file in excel and do the further operations. After loading the

dataset the dataset had columns such as the Entropy, Size, MD5 values, SHA-1 values, DLL characteristics, DLL names, etc. The dataset almost consisted of everything on the basis of which the detection will be performed. We needed only some columns which we collected there are pre-processing of data we needed to do and labeling that had to be done before we proceed with the further steps of detection, the below section will explain pre-processing steps in detail.

3.3 Data Pre-Processing:

The process of Pre-processing of data is very important in machine learning and that is needed to the model that we are using and also it will be necessary to compare it with the other models that we have implemented and compared it to the other model. The feature engineering for our implementation was very easy. The steps taken to do the feature engineering is as explained in detail in the section below.

3.3.1. Data Extraction and conversion:

The final dataset we combined using consists of a total of 56,810 files out of which 35,694 are malicious file and 21,164 are benign files. They all were in .csv file, so there wasn't any need of converting the file from .txt to .csv. The contents of file consisted of several columns such as Size, size of code, DLL characteristics, MD5 values, SHA-a values, etc. out of which we needed only some columns which are DLL characteristics, Entropy, ImageBase, Size of initialized data, Size of uninitialized data and Size, these columns were chosen because these are the main part of the malware that can affect the system and based on this the detection can be done. DLL characteristics are nothing but libraries where it has permissions and other kinds of libraries, with the help of malware these kinds of libraries can be initiated without the user permission. Entropy is nothing but a measurement of randomness required to do code obfuscation etc. Size represents the actual size of the malware where it can be of any size. The above-mentioned features are extracted from the .csv file and we created new .csv files where we combined malicious and benign files.

The above-mentioned detail is all done using Microsoft Excel which is represented in columns and values with the values associated with each of them and we have a total of 56,810 files. To avoid confusion we created a new column were we labeled it an M/B which represents Malware and Benign files.

3.3.2.Feature engineering:

The Dataset that we created after the pre-processing needs to be transformed into a feature matrix so that we can apply it to our model and the rest of the models that we have compared in machine learning. Feature selection is performed because we can get better accuracy and the performance of detection will get better of the malware and benign files. Further, we labeled the dataset and kept it ready, so the next step is to implement the data into the model that we selected and run it.

3.3.2.1. Feature Matrix:

The feature matrix for our dataset was created using 1's and 0's with the help of one-hot coding, this is done to ease out the detection or to ease the input given to the machine and it

understands that easily. As said earlier that we created a dataset combining both malware files and benign files, at the end of the dataset where we created a different column which is 'M/B' which represents malware and benign files. So the malware files are denoted as 1's and the benign files are denoted as 0's, the working is such that if the user inputs as 1 then the output is expected that it is a malware and if the user inputs as 0 then it will give the output as it is a benign file. As of my knowledge, there are no malware datasets with such a feature matrix. So this show we labeled the malware and benign files so that the machine understands and detects it properly. The below diagram represents the labeling and the feature matrix along with it.

A	B	C	D	E	F	G
DllCharacteristics	Entropy	ImageBase	Size	SizeOfInitializedData	SizeOfUninitializedData	M/B
0	5.981249	4194304	76288	2560	1500	1
0	6.081747	4194304	2558464	500348	21476	1
0	5.586422	4194304	178688	177664	33557504	1
0	7.969464	4194304	806816	28672	1355776	1
32768	7.9999	4194304	50689096	38912	110080	1
0	7.328245	4194304	76800	24576	188416	1
0	6.257786	4194304	69660	77824	0	1
0	5.308237	4194304	110146	74240	3072	1

Figure 2: - Dataset creation and labeling

3.3.3. Data Details:

The final dataset that is generated has the following columns and rows which are explained in detail below:

1. The dataset consists of total 56,810 files out of which 35,694 are malicious files and 21,164 are benign files
2. The columns that are in this dataset are DLL characteristics, Entropy, Image Base, Size, SizeOfInitilaizedData, SizeOfUninitializedData, and M/B
3. The M/B column consists of 1's and 0's that are generated for ease of the detection.

The above-mentioned details and the datasets will be applied to different machine learning models and algorithms which will be explained in the below sections.

4.Data Specifications:

The Figure that is shown below shows the process that has been followed for performing the detection of malware and the process that are in the figure is explained in detail below. There are three layers in this process of detecting the malware.

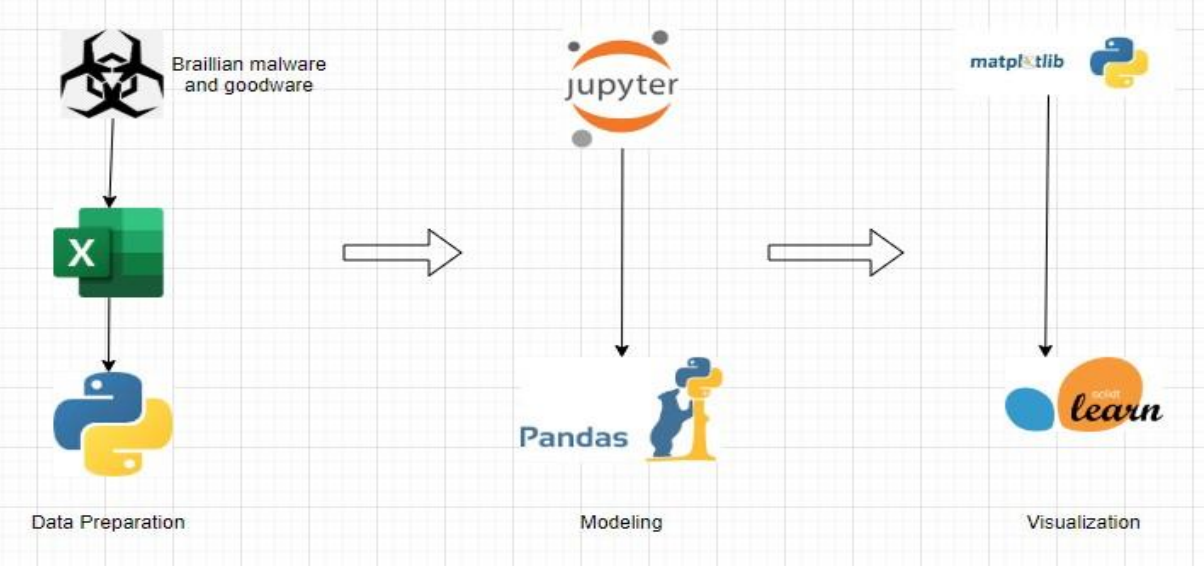


Figure 3: - Design of the implementation

- Data Preparation: - In this step, We collected the dataset of malware and benign application from the website Kaggle[8]. After downloading it we imported it into the excel so that the necessary conversion and data extraction can be performed. Then a combined dataset is prepared to apply it in the different machine learning models.
- Modeling: - In this process different types of machine learning algorithms or models such as Random Forest, Support Vector machines, KNN are implemented. The evaluation was done by observing different parameters such as precision, recall, f1score, and support. These models were implemented in Jupyter Notebook which is an excellent IDE for machine learning and the framework that we imported is the panda.
- Visualization: - The above model was evaluated with the help of the above-mentioned matrix and also confusion matrix is used to see the accuracy and gives a better plotting in a graph and which is inbuilt sckit learn.

The different steps involved with the further implementation of the model along with the evaluation, matrix, and calculations are explained in detail in the below sections.

5.Implementation:

In the implementation step, the dataset that we created was implemented in each machine learning algorithm/model. To achieve the best results and accuracy we performed hyperparameter tuning which is changing of small parameters in the model to achieve better accuracy and performance. Further moving on to the implementation or applying the model we split the dataset into two parts that are 70% of the data is allotted to training and 30% of data is allotted to the test data, this is done because we can test the performance of the data instead of applying the whole data to training. We did the split by applying “train_test_split” which is imported from the “sklearn model” package in python. Explanation of each model

which is the KNN model, SVM model, Random Forest model, and Random forest with decision tree model is explained in detail below.

5.1. Support Vector Machine (SVM) model:

The use of the Support Vector Machine is that it helps to find the hyperplane in an Ndimensional space that classifies the data points. So our main objective in this is to find the maximum margin which means the maximum distance between two data points[9]. Then we apply the dataset in the model which executed as given in the below figure.

```
from sklearn.svm import SVC
svm_clf = SVC(gamma="auto")
svm_clf.fit(X_train, y_train)
```

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

Figure 4: - SVM model creation

5.2. K-nearest neighbors (KNN) model:

The use of the KNN model is that it classifies the data according to the nearest data available. for example. If there are malware and benign files then the malware will go to the class which has the highest number of malware. We do this by setting up parameters such as K-values, Leaf size, and the number of neighbors. The dataset is applied to the model as showed in the figure below.

```
In [25]: from sklearn.neighbors import KNeighborsClassifier
         classifier = KNeighborsClassifier(n_neighbors=50)
         classifier.fit(X_train, y_train)
```

```
Out[25]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                              metric_params=None, n_jobs=None, n_neighbors=50, p=2,
                              weights='uniform')
```

Figure 5:- KNN model creation

5.3. Random Forest:

In random forest has a group of trees in which each tree will classify and give results based on a few attributes. Trees are chosen which has the maximum number of nodes. The model was implemented on the dataset by giving little parameters which are the maximum depth, the estimators, and the criteria. The below figures gives the output.

```
In [9]: from sklearn import model_selection
        from sklearn.ensemble import RandomForestClassifier

In [10]: rfc = RandomForestClassifier()
         rfc.fit(X_train,y_train)

Out[10]: RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                                criterion='gini', max_depth=None, max_features='auto',
                                max_leaf_nodes=None, max_samples=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, n_estimators=100,
                                n_jobs=None, oob_score=False, random_state=None,
                                verbose=0, warm_start=False)
```

Figure 6: - Random Forest model creation

5.4. Random forest with a Decision tree:

Random Forest with decision tree just gives more precision and accuracy but according to the evaluation. It didn't give due to the imbalance in data. The dataset was applied to the mode with some parameters considering the tree size, estimators, and much more.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
import seaborn as sns
```

Figure 7: - Random forest with decision model creation

6.Evaluation:

The evaluation in machine learning is done by calculating metrics by taking the help of a confusion matrix which is in the sklearn metrics package. The graphs and ROC curves are plotted with the help of the confusion matrix for the KNN model and other models that are compared.

When we execute it is considered an important step to calculate the matrix with the help of the fusion matrix. This is in the form of a table or a graph that represents the Accuracy, Precision, F1-score, and Recall of the model that we implemented. The below figure shows how a confusion matrix looks like.

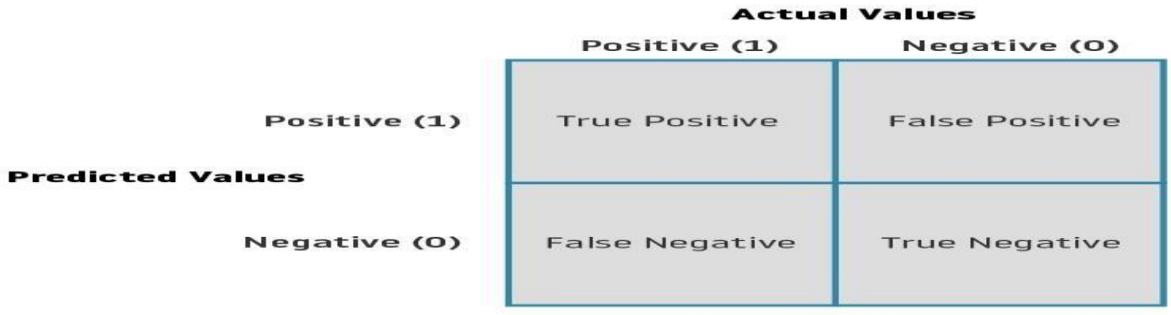


Figure 8: - Design of Confusion matrix

The above mentioned in boxes which are True positives, False positives, False-negative and True negative are explained in detail below.

1. True Positive (TP): True positive means the model or algorithm correctly predicts what has been told to predict. E.g. In our implementation, we have told to detect the malware so the amount of application detected will be shown.
2. False Positive (FP): The meaning of False positive is that it incorrectly predicts the positive that is been told to the algorithm.
3. False Negative (FN): In this, the outcome that is generated represents that the model predicts or detects the negatives incorrectly.
4. True Negative (TN): In this, the model correctly predicts or detects the negatives.

The parameters that are considered in the confusion matrix such as Accuracy, precision, recall, and F1- score are explained below and also the equation that is used to calculate those.

1. Precision: A metric that quantifies the number of correct positive detection made. This calculates the accuracy of the minority class. It is calculated by correctly detected positives divided by the total number of positives detected.

   True Positives / (Trues Positives + False Positives) …… (1)

2. Accuracy: Accuracy means the detected values by the model which are correctly and as told to the model to do. It is calculated by adding TP and TN divided by all the positives and negatives.

   TP + TN / TP + TN + FP + FN…….. (2)

3. F1-Score: F1-Score is a metric in which precision and recall are taken into account. The equation to calculate is as follows:

   2 *(Precision * Recall / Precision + Recall)…… (3)

4. Recall: Recall is the sensitivity or the True positives rate of which the equation is explained below.

   TP / TP + FN…… (4)

From the evaluation and the results that are achieved from the models considered we can say that the KNN algorithm has outperformed other algorithms with a greater percentage of accuracy. Our model can detect a large number of malwares and goodwares and classify them accordingly. Compared to the existing model our model can be trained in large amount where as authors have used the datasets which are less and achieving greater accuracy. We can say that considering the size and amount of malwares and out KNN model has given us a good accuracy which is 78%.

6.1. KNN, SVM, RF, and RF with Decision tree:

After we created the dataset, We applied our dataset into the model or algorithm for training and then obtain the results of them. The confusion matrix for each model was performed to achieve the values of Accuracy, Precision, Recall, and F1-score. According to the values, we decided which model is the best or suitable for the detection of malware. the below figures represent the confusion matrix and the graph plotted for each model.

6.1.1. KNN Model:

The confusion matrix for this model is given below:

```
[[5764  592]
 [2580 3734]]
              precision    recall  f1-score   support

           0       0.69      0.91      0.78      6356
           1       0.86      0.59      0.70      6314

    accuracy                           0.75     12670
   macro avg       0.78      0.75      0.74     12670
weighted avg       0.78      0.75      0.74     12670
```

Figure 9: - Confusion matrix for KNN

The below graph is the Error rate of the KNN model which is decreasing which represents a positive result.

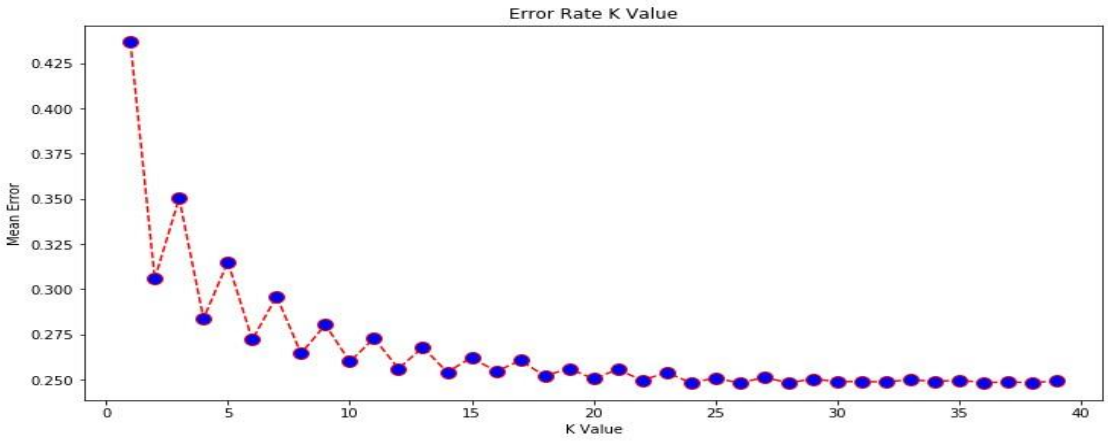


Figure 10: - An error rate of KNN model

6.1.2. Random Forest:

The below figure shows the accuracy rate and the confusion matrix

```
=== Confusion Matrix ===
[[1476 2789]
 [2733 4364]]

=== Classification Report ===
              precision    recall  f1-score   support

           0       0.35      0.35      0.35      4265
           1       0.61      0.61      0.61      7097

    accuracy                           0.51     11362
   macro avg       0.48      0.48      0.48     11362
weighted avg       0.51      0.51      0.51     11362


=== All AUC Scores ===
[0.06097404 0.0576266  0.05750243 0.06309805 0.07044354 0.08775189
 0.13598347 0.11322851 0.21001781 0.25133252 0.19848295 0.26593884
 0.25675389 0.26800981 0.31351806 0.95445192 0.85897155 0.96209692
 0.97510853 0.9650142  0.94914672 0.94838836 0.95262489 0.73652044
 0.59633664 0.27664715 0.24248775 0.32635376 0.79580197 0.93493369
 0.78733389 0.91184602 0.90843257 0.94286411 0.96968716 0.88320589
 0.91026292 0.95442537 0.93463831 0.95357408 0.80080018 0.93969128
 0.9499167  0.93726438 0.75811094 0.88946084 0.79354209 0.8196181
 0.73028923 0.80783424]

=== Mean AUC Score ===
Mean AUC Score - Random Forest:  0.6293669041597464
```

Figure 11: - Confusion matrix for the Random Forest model

6.1.3. SVM Model:

The below figure shows the code and Accuracy rate of the Support vector machine model.

```
from sklearn.model_selection import cross_val_score

svm_scores = cross_val_score(svm_clf, X_train, y_train, cv=10)
svm_scores.mean()
```

```
0.697273663986117
```

Figure 12: - Accuracy for SVM model

6.1.4. Random Forest with Decision tree:

The below diagram shows the confusion matrix and the graph that represents the ROC curve with TP, TN, FP, and FN.

```
=== Classification Report ===
              precision    recall  f1-score   support

           0       0.61      0.63      0.62      6365
           1       0.61      0.60      0.61      6305

    accuracy                           0.61     12670
   macro avg       0.61      0.61      0.61     12670
weighted avg       0.61      0.61      0.61     12670
```

Figure 13: - Confusion matrix for RFDT model

```
Confusion matrix, without normalization
[[3992 2373]
 [2538 3767]]
```

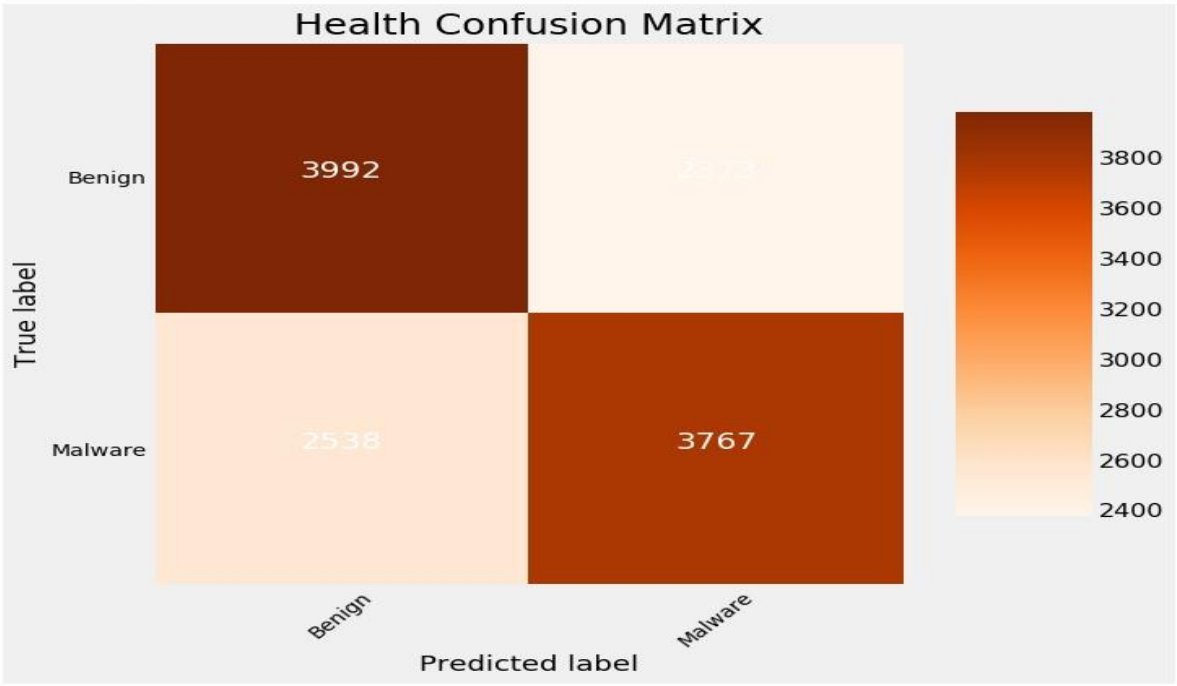
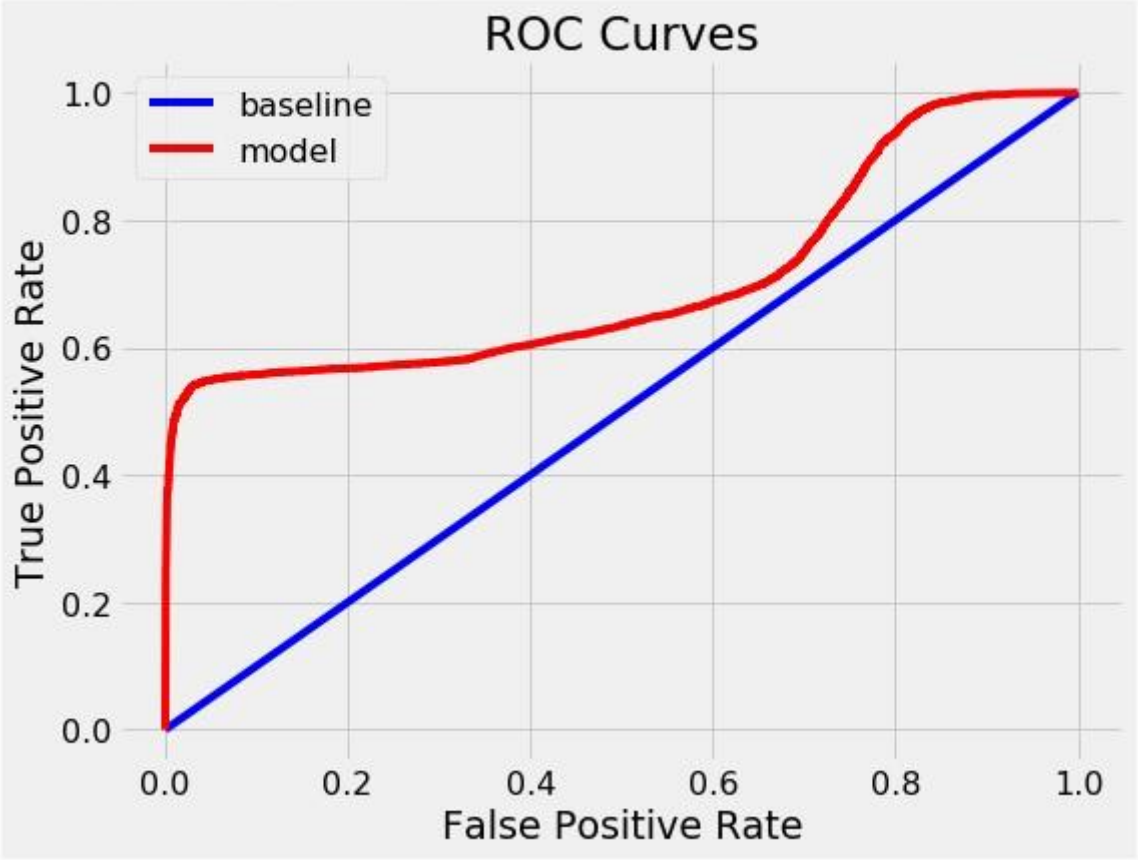


Figure 14: - Graph plotted of the Confusion matrix

The above diagram represents the confusion matrix and shows the amount of detected malware and benign files.

```
Recall Baseline: 1.0 Test: 0.6 Train: 0.77
Precision Baseline: 0.5 Test: 0.61 Train: 0.93
Roc Baseline: 0.5 Test: 0.71 Train: 0.96
```




```
<Figure size 432x288 with 0 Axes>
```

Figure 15: - ROC curve for the model

The above diagram shows the ROC curve in which at some point the line goes decreasing the reason for that to occur is due to there is a slight imbalance in the data.

From the above-represented diagrams and the confusion matrix of each model that is evaluated for achieving the Accuracy, precision, recall, and F1-score. We can observe that the KNN model has achieved the highest accuracy rate which is 78% followed by that Random Forest gives us 51 % of accuracy, SVM model gives us an accuracy rate of 69% and at the last Random Forest with Decision tree gives us an accuracy rate of 61%. The detected rate and the accuracy are at a good rate if compared to the dataset which is on a large scale and also the difference that has occurred between the models and not able to achieve accuracy higher that is because there is a slight imbalance in the data.

7. Discussion:

The reason to implement this paper was to improve the detection of malware on a large scale and with greater accuracy. As compared to the other researchers the dataset that is being

used in this paper is on a large scale. Even though we were not able to achieve the accuracy up to 90% or more than that but it is considered that even with so large dataset we could achieve the accuracy up to 78% which is not bad. We could have bought the accuracy rate at a greater rate but as seen in a graph we can see that there is an imbalance in the dataset.

After we implemented the model we got an accuracy of 78% in the KNN model followed by 51% of Random forest, and second-highest accuracy by SVM which is 69%. We expected that the Random forest with a decision tree will give more accuracy but unfortunately we achieved 61% accuracy.

Limitations: Due to less time we could not able to extract the opcode or install each malware type, analyze or see their behavior, and perform the detection accordingly. Also, we were not able to achieve accuracy at a higher rate. We were not able to implement this into a file or detect the files but yes for sure we have detected the malware and the benign files.

Future Work: In the future, we will work on obtaining the opcode and analyzing the behavior of malware types and try to detect the malware in real-time.

8. Conclusion:

This research paper is proposed and implemented for the detection of the malware and we have successfully detected the malware and also the benign files. Observing the evaluation part we can conclude that the KNN model has outperformed other machine learning algorithms that we have compared in this paper. According to research, the False positive rate should be less which we have achieved by using the KNN model. The Brazilian malware dataset from Kaggle has performed very well which had data on a large scale.

9. Acknowledgment:

At last, I would like to thank my supervisor Prof. Imran Khan who supported me at every stage of the report and guided me with everything with right and excellent knowledge. My supervisor answered every doubt and explained it very nicely. Also, I would like to thank my parents for the love and support for the difficult times I faced during my thesis. I would like to thank my Data analytics friend who helped me with the implementation whenever I got stuck. I would like to thank the National College of Ireland and School of Computing for managing to finish the semesters and also provided the resources that were needed for the thesis even during the Pandemic.

10. References:

[1]. Santosh Joshi, Himanshu Upadhyay, Leonel Lagos, Naga Suryamitra Akkipeddi, Valerie Guerra, "Machine learning Approach for Malware Detection Using Random Forest Classifier on Process List Data Structure".

[2]. I. Shhadat, B. Bataineh, A. Hayajneh, and Z. A. Al-Sharif, "The Use of Machine Learning Techniques to Advance the Detection and Classification of Unknown Malware," *Procedia Comput. Sci.*, vol. 170, pp. 917–922, Jan. 2020, doi: 10.1016/j.procs.2020.03.110.

[3]. D. Gupta and R. Rani, "Improving malware detection using big data and ensemble learning," *Comput. Electr. Eng.*, vol. 86, p. 106729, Sep. 2020, doi: 10.1016/j.compeleceng.2020.106729.

[4] P. Singhal, "Malware Detection Module using Machine Learning Algorithms to Assist in Centralized Security in Enterprise Networks," Int. J. Netw. Secur. Its Appl., vol. 4, no. 1, pp. 61–67, Jan. 2012, doi: 10.5121/ijnsa.2012.4106.

[1] J. Z. Kolter and M. A. Maloof, "Learning to detect malicious executables in the wild," in *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '04*, Seattle, WA, USA, 2004, p. 470, doi: 10.1145/1014052.1014105.

[6] "IMDS." https://dl.acm.org/doi/pdf/10.1145/1281192.1281308 (accessed Apr. 05, 2020).

[7] C.P. Patidar, Harshita Khandelwal, Zero Day Attack Detection Using Machine Learning Techniques. jrar.org. (2020). [online] Available at: http://www.ijrar.org/papers/IJRAR19J1648.pdf [Accessed 12 Feb. 2020].

[8]"Malware Goodware Dataset." https://kaggle.com/arbazkhan971/malware-goodware-dataset (accessed Aug. 16, 2020).

[9] Chen, Y., Narayanan, A., Pang, S., Tao, B., 2012. Multiple sequence alignment and artificial neural
networks for malicious software detection. 2012 8th International Conference on Natural Computation, (Icnc),pp.261–265.Availableat: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6234576.

[10] "IEEE Xplore Full-Text PDF:" https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6008141 (accessed Apr. 05, 2020).

[11] "IEEE Xplore Full-Text PDF:" https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8724096 (accessed Apr. 05, 2020).

[12] "IEEE Xplore Full-Text PDF:" https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8889540 (accessed Apr. 05, 2020)

[13] "Practical real-time intrusion detection using machine learning approaches | Elsevier Enhanced Reader." https://reader.elsevier.com/reader/sd/pii/S014036641100209X?token=F986553B2DDE89A6822A4BC9F141E8789146FF865C0A7AF114DDB8EDA3BF67EEC558013EFC45852594C573BF79B7231B (accessed Apr. 05, 2020).

[14] Z. Li et al., "VulDeePecker: A Deep Learning-Based System for Vulnerability Detection," Proc. 2018 Netw. Distrib. Syst. Secur. Symp., 2018, doi: 10.14722/ndss.2018.23158.

[15]“K-Nearest Neighbors Algorithm in Python and Scikit-Learn,” *Stack Abuse*. https://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/ (accessed Aug. 16, 2020).

[16].R. Gandhi, “Support Vector Machine — Introduction to Machine Learning Algorithms,” *Medium*, Jul. 05, 2018. https://towardsdatascience.com/support-vector-machine-introduction-to-machinelearning-algorithms-934a444fca47 (accessed Aug. 16, 2020).

[17]J. Huneycutt, “Implementing a Random Forest Classification Model in Python,” *Medium*, May 21, 2018. https://medium.com/@hjhuney/implementing-a-random-forest-classification-model-inpython-583891c99652 (accessed Aug. 16, 2020).

[18]“Random Forests Classifiers in Python,” *DataCamp Community*, May 16, 2018. https://www.datacamp.com/community/tutorials/random-forests-classifier-python (accessed Aug. 16, 2020).

[19]“Crisp DM methodology,” *Smart Vision Europe*. https://sv-europe.com/crisp-dm-methodology/ (accessed Aug. 16, 2020).

[20] H. N. B, “Confusion Matrix, Accuracy, Precision, Recall, F1 Score,” *Medium*, Jun. 01, 2020. https://medium.com/analytics-vidhya/confusion-matrix-accuracy-precision-recall-f1-scoreade299cf63cd (accessed Aug. 16, 2020).

[21] J. Brownlee, “How to Calculate Precision, Recall, and F-Measure for Imbalanced Classification,” *Machine Learning Mastery*, Jan. 02, 2020. https://machinelearningmastery.com/precision-recalland-f-measure-for-imbalanced-classification/ (accessed Aug. 16, 2020).