MALICIOUS ANDROID APPLICATION DETECTION USING MACHINE LEARNING

Project report submitted in partial fulfilment of the requirement for the degree of
Bachelor of Technology

In

Computer Science and Engineering/Information Technology

By

Aditya Kapoor (151359)

Himanshu Kushwaha (151361)

Under the supervision of

Dr. Ekta Gandotra

Department of Computer Science & Engineering and Information Technology

**Jaypee University of Information Technology Waknaghat,
Solan-173234, Himachal Pradesh**

Candidate's Declaration

We hereby declare that the work presented in this report entitled **"MALICIOUS ANDROID APPLICATION DETECTION USING MACHINE LEARNING"** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of our own work carried out over a period from August 2018 to December 2018 under the supervision of **Dr. Ekta Gandotra** (Assistant Professor(Senior Grade), Computer Science and Engineering And Information Technology)

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Aditya Kapoor, 151359

Himanshu Kushwaha, 151361

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Dr. Ekta Gandotra
Assistant Professor (Senior Grade)
Computer Science and Engineering and Information Technology
Dated:

Acknowledgement

We would like to express our special thanks of gratitude to our teacher and mentor **Dr. Ekta Gandotra** who gave us the golden opportunity to do this project on the topic **MALICIOUS ANDROID APPLICATION DETECTION USING MACHINE LEARNING**, which also helped us in doing a lot of Research and we came to know about so many new things. We are really thankful to her.

Secondly, we would also like to thank Lab assistants who helped us a lot in finalizing this project within the limited time frame.

Aditya Kapoor (151359)

Himanshu Kushwaha (151361)

Table of Content

Publications

[1] A. Kapoor, H. Kushwaha, E. Gandotra, "Permission Based Android Malicious Application Detection Using Machine Learning". In the proceedings of 5th International Conference on Signal Processing and Communication. (ICSC- 2019), **[Accepted and Presented]**

List of Abbreviations

- ML- Machine learning
- AI- Artificial Intelligence
- LR- Logistic Regreesion
- LDA- Linear Discriminant Analysis
- KNN- K nearest neighbors
- SVM- Support Vector Machines
- NB- Naïve Bayes
- APK- Android Package Kit
- CSV- Comma Separated Values
- OS –Operating Systems
- URL- Uniform Resource Locator
- CNN- Convolutional neural network
- DL- Deep Learning

List of Figures

List of Graphs

List of Tables

Abstract

Since the launch of the smartphones, their usage is increasing exponentially and it has become an important part of our lives. We are very much dependent on smartphones for our daily routine and use numerous applications both from the play store or the third party applications. Most of the times the applications downloaded from unofficial sources pose a threat as it doesn’t undergoes the necessary checks or mechanisms to validate the authenticity of these applications and maybe infected with malware. The malware infected applications can lead to leakage of user’s personal data or for getting restricted access to the system. Initially, the use of signatures, which are a small number of bytes from the virus, were carried out to check the viruses but its database needs to be updated regularly. In this project, we present an alternative of virus detection by using machine learning techniques we extracted the permissions and created a dataset and used machine learning algorithms for classifying the applications into malicious or benign and compared their results to determine the best algorithm suiting for our dataset. Furthermore, we have converted the Android application samples into images and explored how convolutional neural network works for the classification of application into malicious or benign.

Aims and Objectives

The aim of the project is to enhance the security measures of the Android application and to protect the users from the spread of viruses and the leakage of personal data or the unauthorized access to the restricted sections. This is done by using permissions and classifying the android applications into malicious or benign on the basis of the permissions they use. It also speeds up the malware detection process and performs better than the traditional method of virus detection- signatures.

Objectives-

- Study about various malwares and its effects on the android system. And also about the permissions requested by Android application and also about various machine learning algorithms and their differences.
- Download multiple android application samples both malicious and benign to create a dataset to train the model. And creation of script to extract the permissions from apk samples and list them into CSV file.
- Application of various Machine Learning algorithms to the created dataset and comparing the algorithms on the basis of parameters like accuracy, precision etc.
- Conversion of Android samples into images. And Implementation of CNN on the images and classification into benign and malicious.

Chapter-1

INTRODUCTION

(1.1) Introduction to the Project

In today’s era, smartphones have become a ubiquitous device for storage, computing and for carrying out the transactions among these devices. It is more handy, portable and easy to use device. To make the usage experience of the mobile phones better and for determining the features and functions available on the device, a software platform i.e. an operating system is preinstalled in the mobile phones which is decided by the manufacturer. This operating system is specially designed for mobile phones and largely vary from the operating systems of that of computers and therefore able to run advanced functions to smartphones that were previously unable to be done on desktop computers. There are various kinds of operating systems available in the market namely Android OS, IOS, Windows OS etc. The first fully functional and popular smartphone operating system was the Symbian OS which was introduced in 2000. Another operating system that revolutionized the market was the IOS by Apple and came along with their first iphone model in 2007. But since the launch of Android operating system in 2007, it has become the most popular mobile operating system and has grown strongly through the years. According to the **statista** report on sales share globally in mobile OS market to end users from 2009 to 2018, 88% percent of all smartphones that were sold to the customers or the end users had Android OS in them. The availability of smartphones with android OS at a relatively cheaper rates have also led to this accelerated migration of feature phone users to smartphone users and exponential growth in android market.

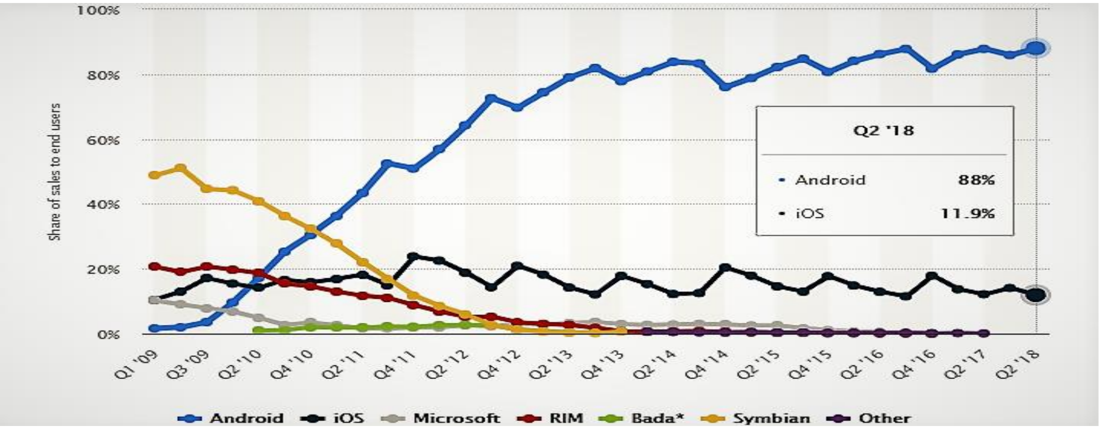


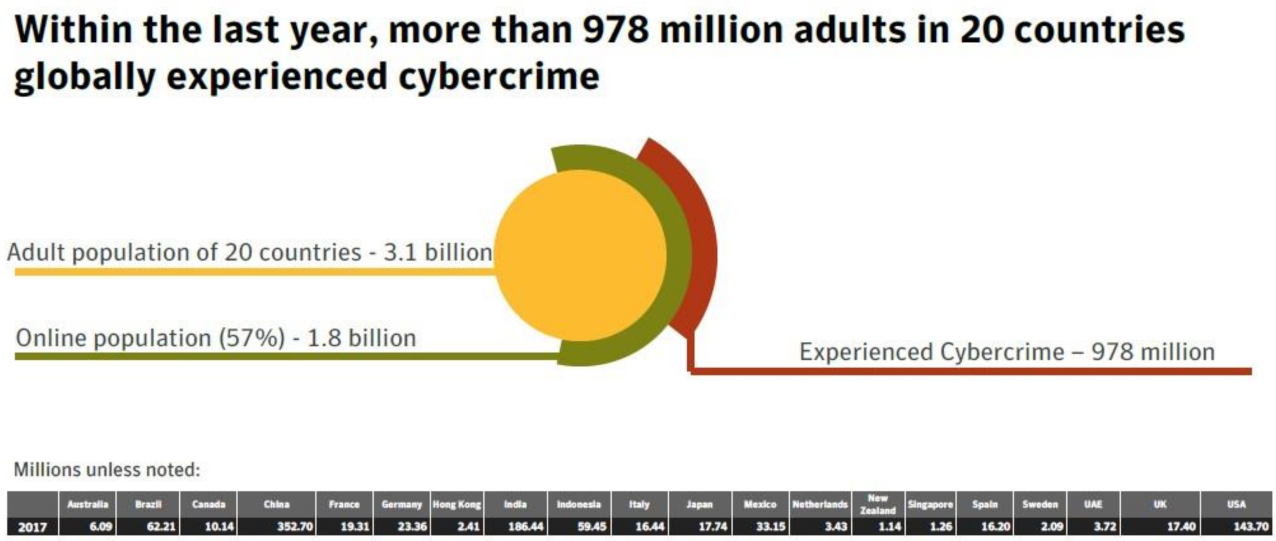
Fig 1.1: Statista report on sales share globally in mobile OS market

This accelerated growth of the android OS has largely attracted the malware developers and a large number of malware containing applications are being developed every day. Smartphones are becoming a major target to malware attacks with Android OS being the top on the hit list as it is open source OS and it is relatively easy to penetrate malware and viruses in these applications. Thus, the detection and removal of these malware containing applications and is becoming a major concern for both developers and to the end users. It is becoming the need of the hour to keep the platform safe for the community by providing detection and defensive methods against malware.

(1.2) MOTIVATION

In this project we work towards the improvement of user experience and the security measures of the android community. As discussed above a large proportion of the population rely on their smartphones for their daily transactions and thus need to be very secure and confidential for gaining trust of user and stand onto the expectations of the users. Most of the customers are overconfident when it comes to the security prowess and often keep their smartphones in vulnerable conditions and thus make it easier for the cybercriminals to attack the system and become able to steal data or leak personal data.

According to the global report on Cyber Security Insights submitted by Norton in 2017, there has been 978 million people collective in 20 countries which were affected by cybercrime and as a result of this cybercrime, victims lost a collective of 172 billion US dollars globally and the average loss of 142 dollars per person which is huge.

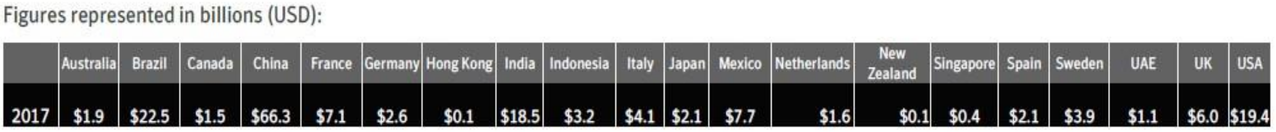


Millions unless noted:

	Australia	Brazil	Canada	China	France	Germany	Hong Kong	India	Indonesia	Italy	Japan	Mexico	Netherlands	New Zealand	Singapore	Spain	Sweden	UAE	UK	USA
2017	6.09	62.21	10.14	352.70	19.31	23.36	2.41	186.44	59.45	16.44	17.74	33.15	3.43	1.14	1.26	16.20	2.09	3.72	17.40	143.70

Fig 1.2: Analysis by Norton about Cybercrimes.

And the distributed representation of loss per person according to the country-



Figures represented in billions (USD):

	Australia	Brazil	Canada	China	France	Germany	Hong Kong	India	Indonesia	Italy	Japan	Mexico	Netherlands	New Zealand	Singapore	Spain	Sweden	UAE	UK	USA
2017	$1.9	$22.5	$1.5	$66.3	$7.1	$2.6	$0.1	$18.5	$3.2	$4.1	$2.1	$7.7	$1.6	$0.1	$0.4	$2.1	$3.9	$1.1	$6.0	$19.4

According to the same report by Norton, a survey conducted by the Norton to enumerate the people according to the awareness and how careful they are when it comes to storing passwords-

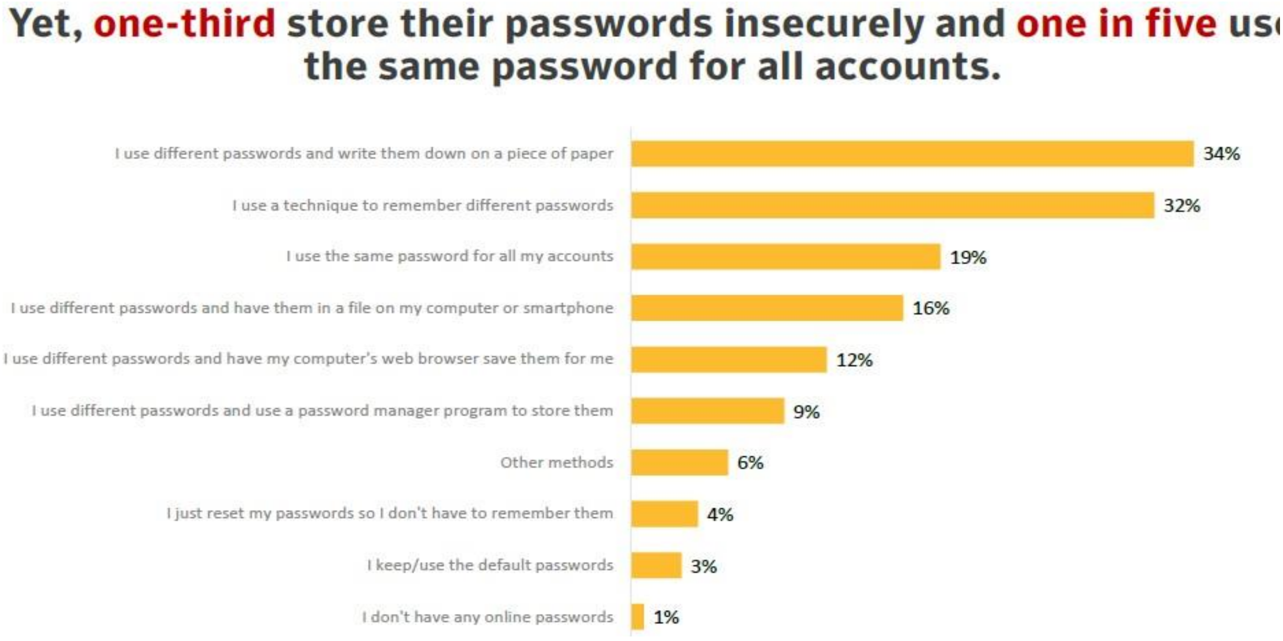


Fig 1.3: Analysis by Norton about how aware users are about Security.

Thus, a graph was made to show the comparative analysis and to identify the area which needs to be targeted for improving the security measures. And the result was-

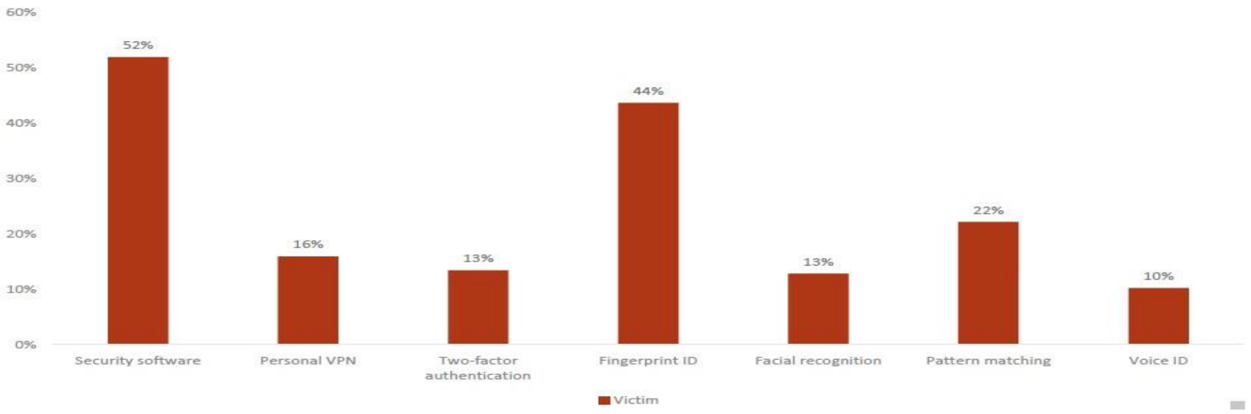


Fig 1.4: Graphical representation of distribution of victims of cyberattacks by methods.

(1.3) Traditional Approaches

The steep rising of the malware have drawn the attention of the researchers towards the malware detection techniques. Malware detectors are the tools which are used by the developers and the android community to check the application for malware. The quality and efficiency of such detectors are measured by certain techniques they employ. Many intrusive detection methods which were used can be listed into three categories namely host-based, cloud based or social collaboration. In the case of host-based method, the detection process runs entirely in mobile device, it assumes that the capabilities of smartphones increases following Moore's theory. This method has an advantage that it will relocate some of the server functionality to the client-side and this will further result in the reduction of communication latencies while the cost in terms of the use of bandwidth is eliminated with ability for real-time detection. This method has a slight disadvantage that as it poses difficulty during the implementation process because of resource-poor limitations of mobile devices.

The malware detection methods can be broadly classifies into three main categories – Signature based detection, anomaly- based detection which can be also referred to as Behaviour based detection and virtual Machine based.

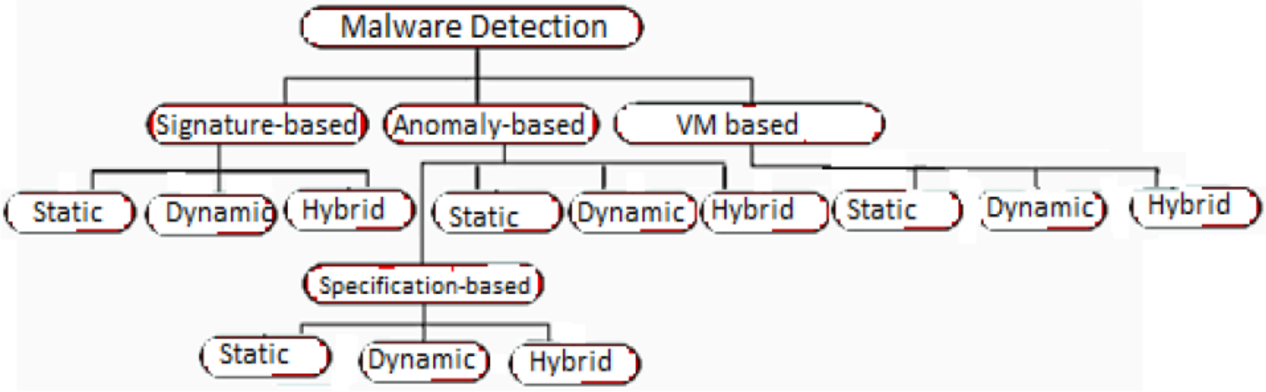


Fig 1.5: Methods of malware detection.

All the detection methods are further classified into static, dynamic and hybrid. In the case of anomaly based or signature based technique, the approach is determined by the method of collecting information to be used for malware detection.

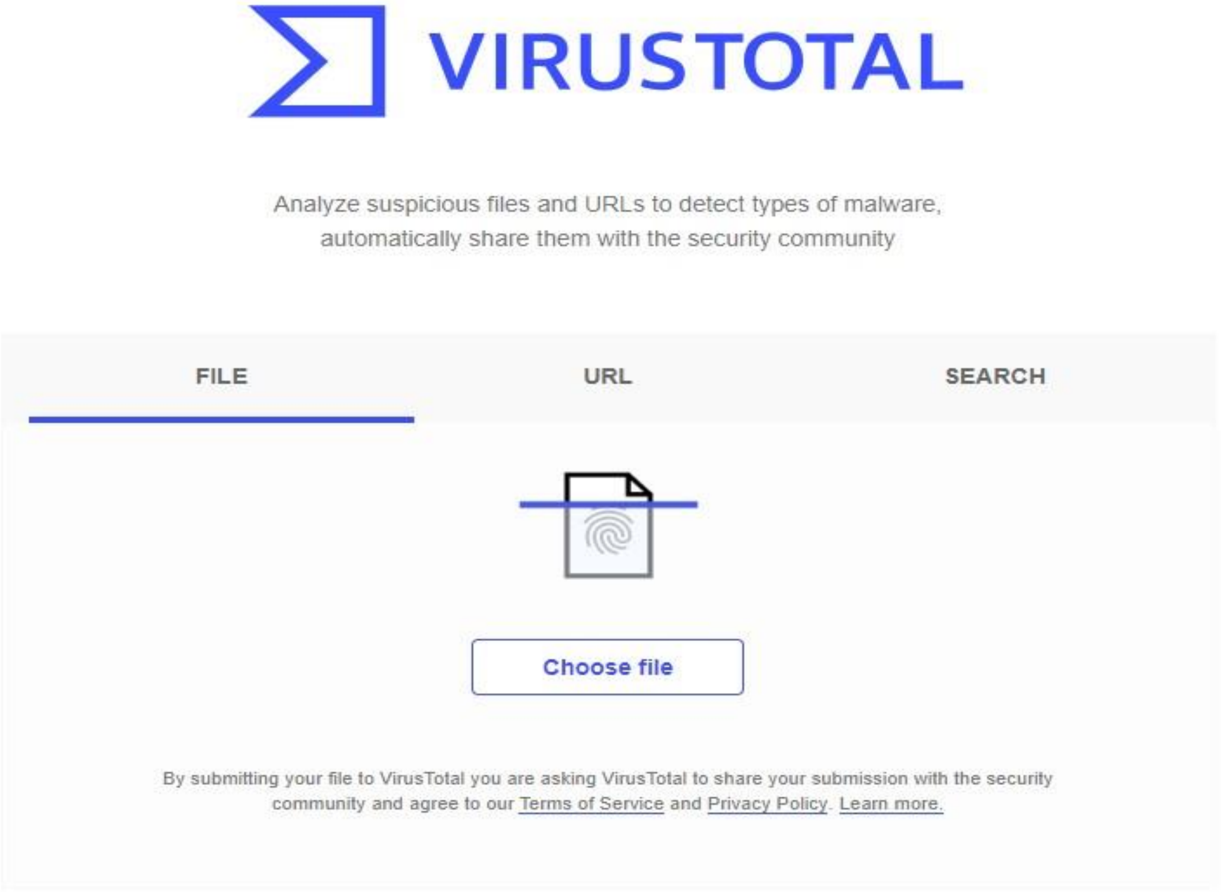
The systems which use the anomaly based malware detection requires a training phase before the detection of malware to determine the normal behaviour of the system and for comparison. In this case, the system which is used for detection of malware is trained on the normal behaviour which makes it easier to differentiate the anomalous behaviour from the normal behaviour occurring in the system. This technique can also be used to detect new and unknown attacks by malware. This method requires the use of feature vectors for training the classifier before the detection and classification of the sample is carried out.

But this method have its disadvantages too. It was clearly shown by a paper by Didier Stevens that if a malware script is zero padded with a certain amount of zero bytes, then it becomes difficult and sometimes remains undetectable by the virus scanners used by him. [1]

Further, a site consisting 60+ anti-virus scanners and aggregates many antivirus products and online search engines and check whether the file uploaded to the site is infected or not and even determines the viruses missed out by the user's antivirus or listed as false positive. Another drawback of signature based approach is that the source code can be manipulated and can be rearranged to avoid the occurrence of common signature which makes it difficult for the anti-virus to collect and create a signature for that sample.

Even in the case of polymorphic and metamorphic viruses, the process of creating signature becomes difficult. It is because in polymorphic viruses, a decrypter/ encrypter stub is used and a different key is used every time the virus is transferred to a different system and thus the body of the virus is different. On the other hand, in the case of metamorphic viruses, the virus is able to evolve themselves and can re-code themselves effectively.

Metamorphic also involves techniques that can substitute the machine code registers used for alternating or inverting the logic of statements



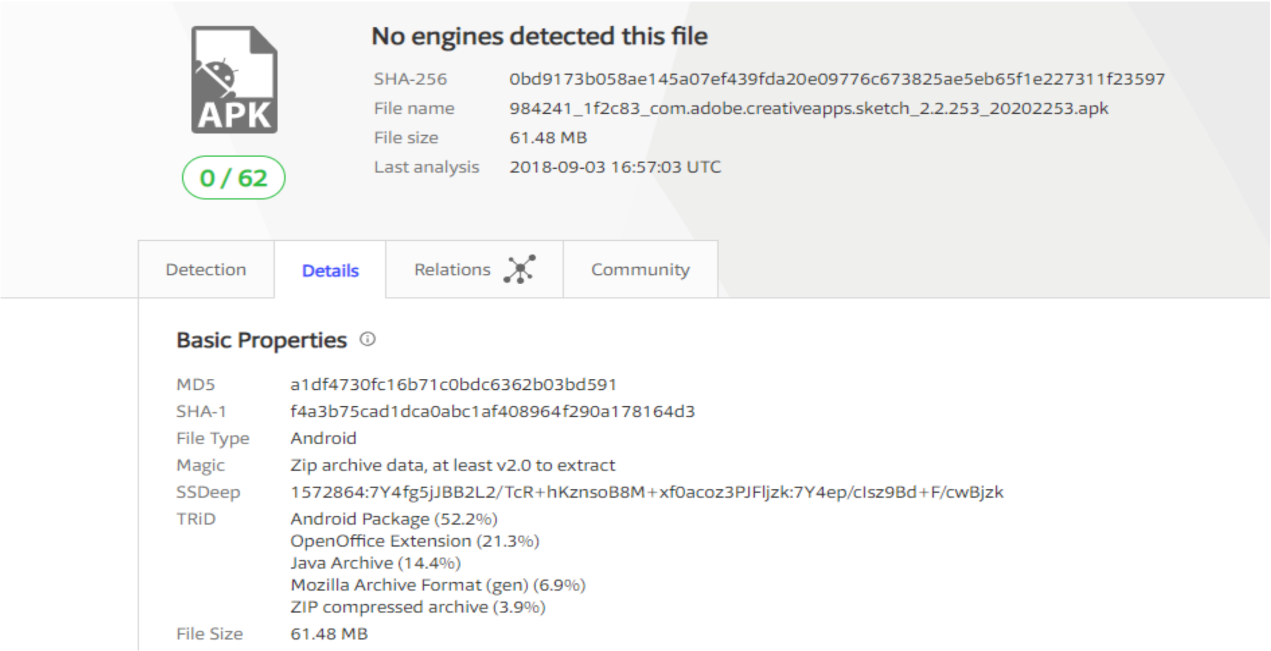
Analyze suspicious files and URLs to detect types of malware, automatically share them with the security community

By submitting your file to VirusTotal you are asking VirusTotal to share your submission with the security community and agree to our Terms of Service and Privacy Policy. Learn more.

Fig 1.6: VirusTotal Webpage

VirusTotal is an online website providing free services used for the analysis of files and URLs for any kind of malicious content. Currently, it inspects the files with its 69 antivirus scanners for the detection of malicious content and give the detailed information about the file. It is freely available of the non-commercial users.

Below we have inspected some files to show how the website works:



No engines detected this file

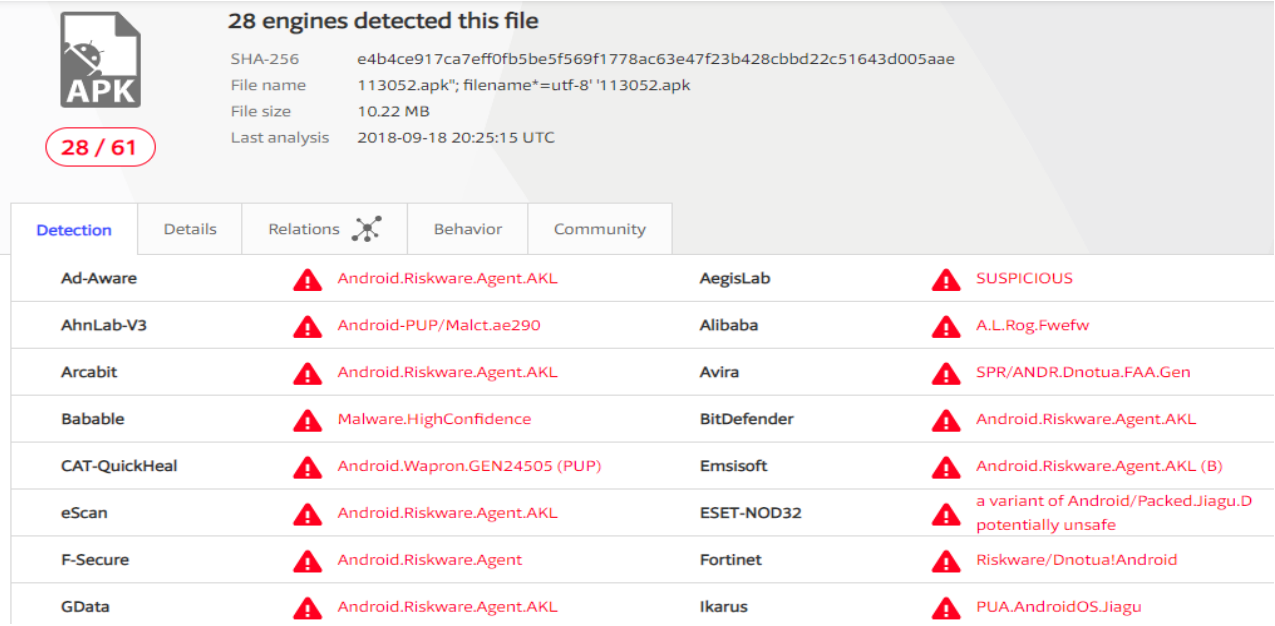
SHA-256	0bd9173b058ae145a07ef439fda20e09776c673825ae5eb65f1e227311f23597
File name	984241_1f2c83_com.adobe.creativeapps.sketch_2.2.253_20202253.apk
File size	61.48 MB
Last analysis	2018-09-03 16:57:03 UTC

Basic Properties

MD5	a1df4730fc16b71c0bdc6362b03bd591
SHA-1	f4a3b75cad1dca0abc1af408964f290a178164d3
File Type	Android
Magic	Zip archive data, at least v2.0 to extract
SSDeep	1572864:7Y4fg5jJBB2L2/TcR+hKznsoB8M+xf0acoz3PJFljzk:7Y4ep/cIsz9Bd+F/cwBjzk
TRiD	Android Package (52.2%) OpenOffice Extension (21.3%) Java Archive (14.4%) Mozilla Archive Format (gen) (6.9%) ZIP compressed archive (3.9%)
File Size	61.48 MB

Fig 1.7: VirusTotal Result on benign android sample.

This is the snapshot of a non-malicious file scanned with the VirusTotal.



28 engines detected this file

SHA-256	e4b4ce917ca7eff0fb5be5f569f1778ac63e47f23b428cbbd22c51643d005aae
File name	113052.apk"; filename*=utf-8' '113052.apk
File size	10.22 MB
Last analysis	2018-09-18 20:25:15 UTC

Ad-Aware	Android.Riskware.Agent.AKL	AegisLab	SUSPICIOUS
AhnLab-V3	Android-PUP/Malct.ae290	Alibaba	A.L.Rog.Fwefw
Arcabit	Android.Riskware.Agent.AKL	Avira	SPR/ANDR.Dnotua.FAA.Gen
Babable	Malware.HighConfidence	BitDefender	Android.Riskware.Agent.AKL
CAT-QuickHeal	Android.Wapron.GEN24505 (PUP)	Emsisoft	Android.Riskware.Agent.AKL (B)
eScan	Android.Riskware.Agent.AKL	ESET-NOD32	a variant of Android/Packed.Jiagu.D potentially unsafe
F-Secure	Android.Riskware.Agent	Fortinet	Riskware/Dnotua!Android
GData	Android.Riskware.Agent.AKL	Ikarus	PUA.AndroidOS.Jiagu

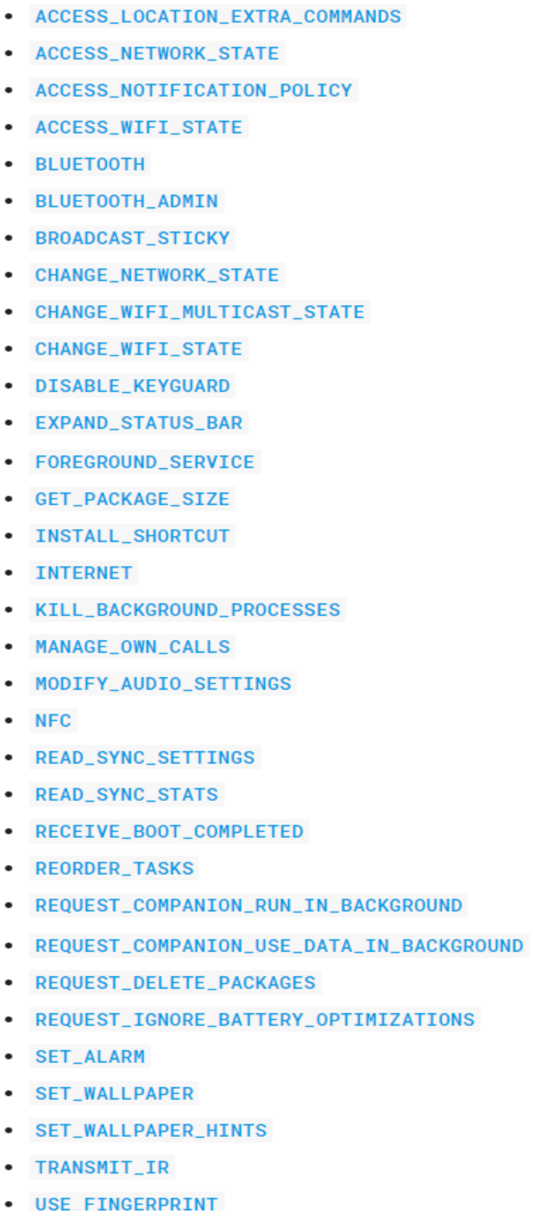
Fig 1.8: VirusTotal report on malicious android sample.

This is the snapshot of a malicious file scanned with the VirusTotal.

Here in this project we are going to talk about android permissions, Android permissions are divided in to several protection levels:

1. Normal Permissions are those permissions which have very little risk of user’s privacy. These permissions do not require user’s involvement; these are granted by the android system directly.

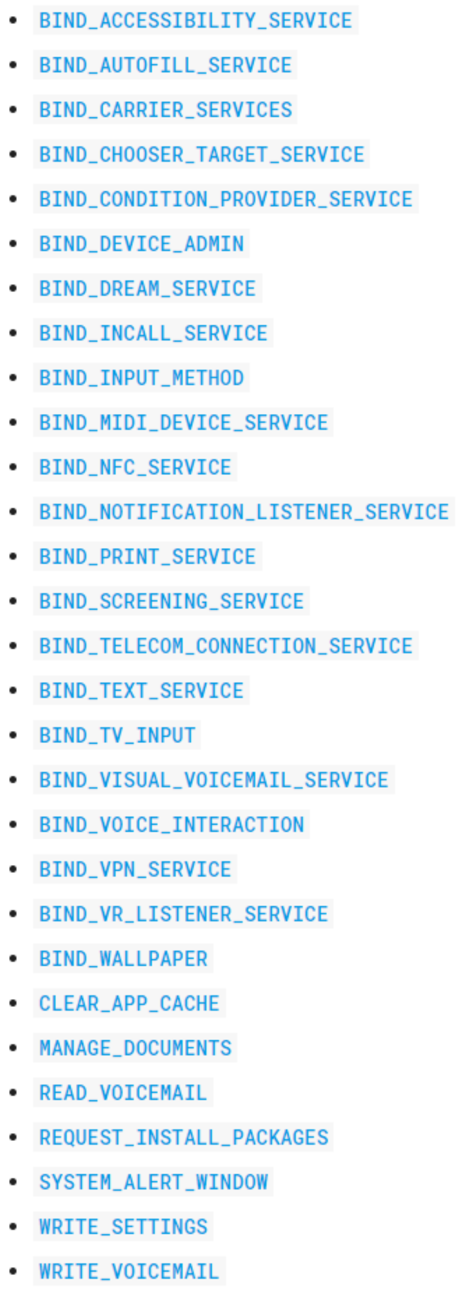
Following permissions comes under the PROTECTION_NORMAL.



- ACCESS_LOCATION_EXTRA_COMMANDS
- ACCESS_NETWORK_STATE
- ACCESS_NOTIFICATION_POLICY
- ACCESS_WIFI_STATE
- BLUETOOTH
- BLUETOOTH_ADMIN
- BROADCAST_STICKY
- CHANGE_NETWORK_STATE
- CHANGE_WIFI_MULTICAST_STATE
- CHANGE_WIFI_STATE
- DISABLE_KEYGUARD
- EXPAND_STATUS_BAR
- FOREGROUND_SERVICE
- GET_PACKAGE_SIZE
- INSTALL_SHORTCUT
- INTERNET
- KILL_BACKGROUND_PROCESSES
- MANAGE_OWN_CALLS
- MODIFY_AUDIO_SETTINGS
- NFC
- READ_SYNC_SETTINGS
- READ_SYNC_STATS
- RECEIVE_BOOT_COMPLETED
- REORDER_TASKS
- REQUEST_COMPANION_RUN_IN_BACKGROUND
- REQUEST_COMPANION_USE_DATA_IN_BACKGROUND
- REQUEST_DELETE_PACKAGES
- REQUEST_IGNORE_BATTERY_OPTIMIZATIONS
- SET_ALARM
- SET_WALLPAPER
- SET_WALLPAPER_HINTS
- TRANSMIT_IR
- USE FINGERPRINT

Fig 1.9: Permissions listed under PROTECTION_NORMAL

2. Signature Permissions



- BIND_ACCESSIBILITY_SERVICE
- BIND_AUTOFILL_SERVICE
- BIND_CARRIER_SERVICES
- BIND_CHOOSER_TARGET_SERVICE
- BIND_CONDITION_PROVIDER_SERVICE
- BIND_DEVICE_ADMIN
- BIND_DREAM_SERVICE
- BIND_INCALL_SERVICE
- BIND_INPUT_METHOD
- BIND_MIDI_DEVICE_SERVICE
- BIND_NFC_SERVICE
- BIND_NOTIFICATION_LISTENER_SERVICE
- BIND_PRINT_SERVICE
- BIND_SCREENING_SERVICE
- BIND_TELECOM_CONNECTION_SERVICE
- BIND_TEXT_SERVICE
- BIND_TV_INPUT
- BIND_VISUAL_VOICEMAIL_SERVICE
- BIND_VOICE_INTERACTION
- BIND_VPN_SERVICE
- BIND_VR_LISTENER_SERVICE
- BIND_WALLPAPER
- CLEAR_APP_CACHE
- MANAGE_DOCUMENTS
- READ_VOICEMAIL
- REQUEST_INSTALL_PACKAGES
- SYSTEM_ALERT_WINDOW
- WRITE_SETTINGS
- WRITE_VOICEMAIL

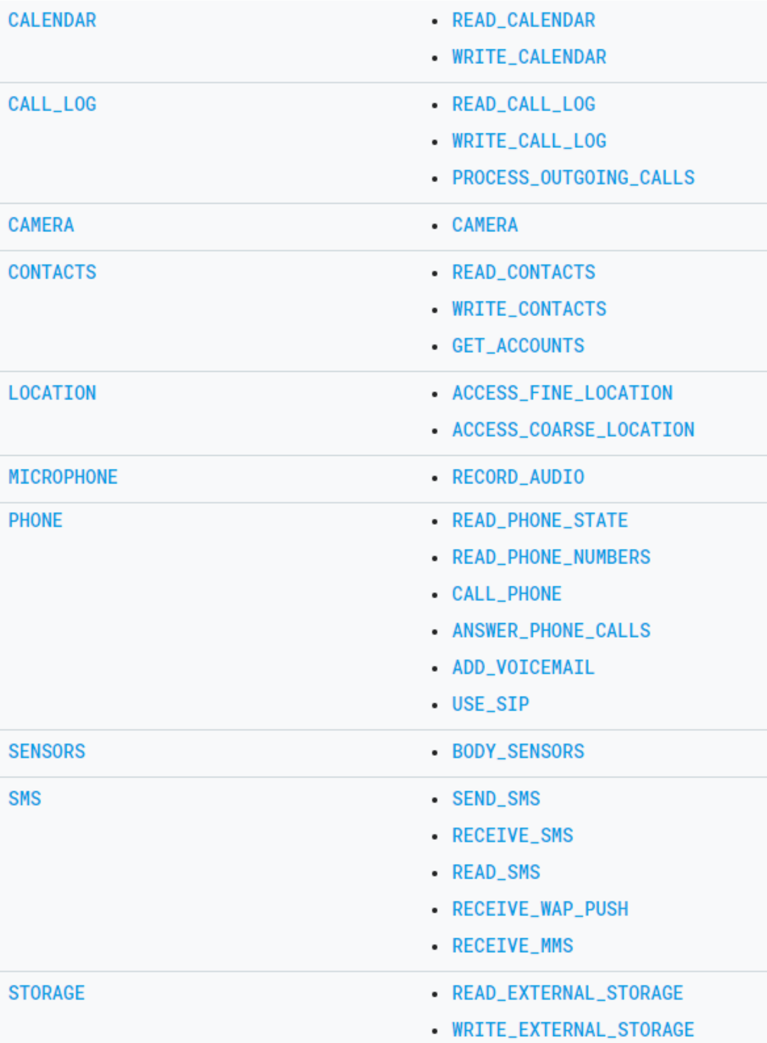
Fig 1.10: Permissions listed under signature permissions.

3. Special Permissions

The permissions that doesn’t comes under the normal and dangerous are the special permissions SYSTEM_ALERT_WINDOW and WRITE_SETTINGS are particularly sensitive, if an application wants to access these it must declare it in the manifest and access those with the help of intents.

4. Dangerous permissions

   These are the permissions which require access to user's private data. For granting these permissions a message is prompt on the screen asking about the user's permissions.



CALENDAR	• READ_CALENDAR • WRITE_CALENDAR
CALL_LOG	• READ_CALL_LOG • WRITE_CALL_LOG • PROCESS_OUTGOING_CALLS
CAMERA	• CAMERA
CONTACTS	• READ_CONTACTS • WRITE_CONTACTS • GET_ACCOUNTS
LOCATION	• ACCESS_FINE_LOCATION • ACCESS_COARSE_LOCATION
MICROPHONE	• RECORD_AUDIO
PHONE	• READ_PHONE_STATE • READ_PHONE_NUMBERS • CALL_PHONE • ANSWER_PHONE_CALLS • ADD_VOICEMAIL • USE_SIP
SENSORS	• BODY_SENSORS
SMS	• SEND_SMS • RECEIVE_SMS • READ_SMS • RECEIVE_WAP_PUSH • RECEIVE_MMS
STORAGE	• READ_EXTERNAL_STORAGE • WRITE_EXTERNAL_STORAGE

Fig 1.11: Permissions listed under dangerous permissions

(1.4) Methodology

(a) **Identifying Malware Samples**

As previously mentioned in the traditional approaches section that most of the prevailing anti-virus scanners use the signature based approach. But there are many viruses that remain unnoticed by these anti-viruses and are discovered after about 3-4 months of entering the system when the system is very much affected. Thus, there is a clear need to find a more effective and fast solution for the aforementioned problems and diminish the evolution of malware in the android system.

(b) **Using Java based Android package analyser**

Thus, in this project, we use an approach which is aimed at uncovering the already known malware families and also the unknown malware to reduce chances of malware in the android community from escaping detection from scanners. For this we created a dataset using multiple android .apk samples downloaded from both google play and VirusShare and other trusted sites providing malware samples. We got a collection of about 4000+ samples of both malicious and benign android application samples. As the permissions required by a particular application is inside the android manifest file of the android sample. There are multiple methods to extract the methods from the android sample. We accessed the permissions requested by a particular android application sample from the manifest.xml file of the android sample using a JAVA program to extract the permissions and store it.

This JAVA decompiler program was developed by self by reverse engineering the JAVA decompiler tools available in the internet for the extraction of permission and other necessary details which were irrelevant to our project and was thus increasing the space and time required for extracting permissions.

(c) **Using Python based Android package analyser**

But this process was a cumbersome process and took very long time for extraction of permissions for a huge data. Thus, we created a script in python which reads and processes multiple samples at the same time and accesses the manifest.xml file and extract permissions and compile the permissions into a CSV format file which could be further used in the machine learning algorithms. This python script was run multiple times on the same samples to ensure the correct data and to lower the chances of randomization. The final result is used as a dataset in our project and made free from error and stored.

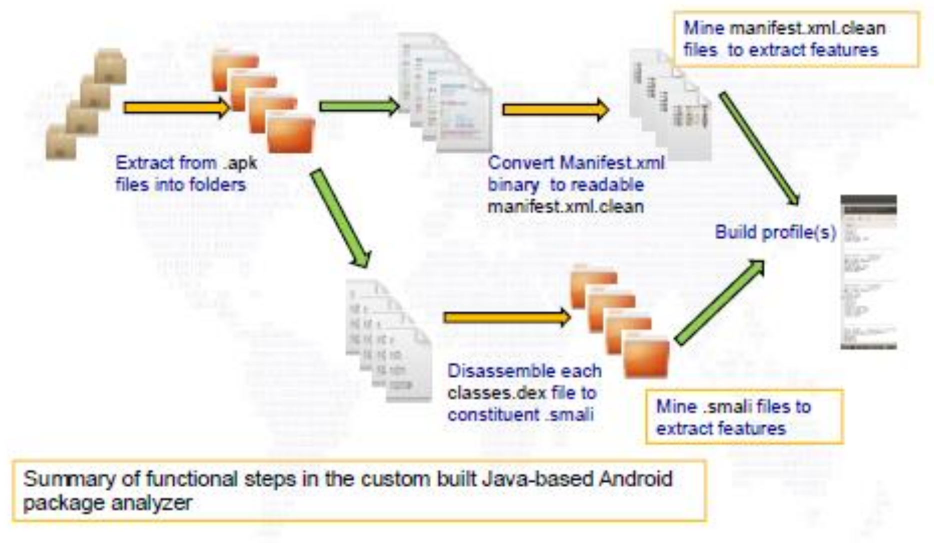

Mine manifest.xml.clean files to extract features
Extract from .apk files into folders
Convert Manifest.xml binary to readable manifest.xml.clean
Build profile(s)
Disassemble each classes.dex file to constituent .smali
Mine .smali files to extract features
Summary of functional steps in the custom built Java-based Android package analyzer

Fig 1.12: Steps followed for building Android package analyser based on Java.

(d) **Using machine learning**

After the creation of the dataset, a script is run to analyse the permissions and classify the samples into two different classes i.e. – malicious and benign. In this project we focus on creating a labelled dataset for the machine learning algorithms and specifically for the supervised learning algorithms. The classifier is then trained using the previously created dataset and then tested to predict the result according to the feature vectors. In using machine learning, we encountered two main problems, at first the need to extract some variety of feature illustration of the applications and second, the requirement for a knowledge set that's virtually completely benign or well labelled which is able to be wont to train a classifier. In addressing these issues, 1st a heterogeneous knowledge square measure extracted processed and vectored into a feature set. Secondly, a K-NN classifier is trained victimisation a normality model that describes to the classifier the traditional behaviour. With this, it becomes doable to find abnormal behaviour by trying to find behaviours that deviate from the outlined traditional behaviours of applications.

Another approach that can be followed is the use of novelty classifier, in which there is requirement of only one class. In this case, the system which is trained needs only one class for training with normal system activity which makes it possible to infer the abnormal activity.

No matter how complex or how advanced the machine algorithm is, any approach of the machine learning used can never be fully efficient to prevent the transmission of viruses.

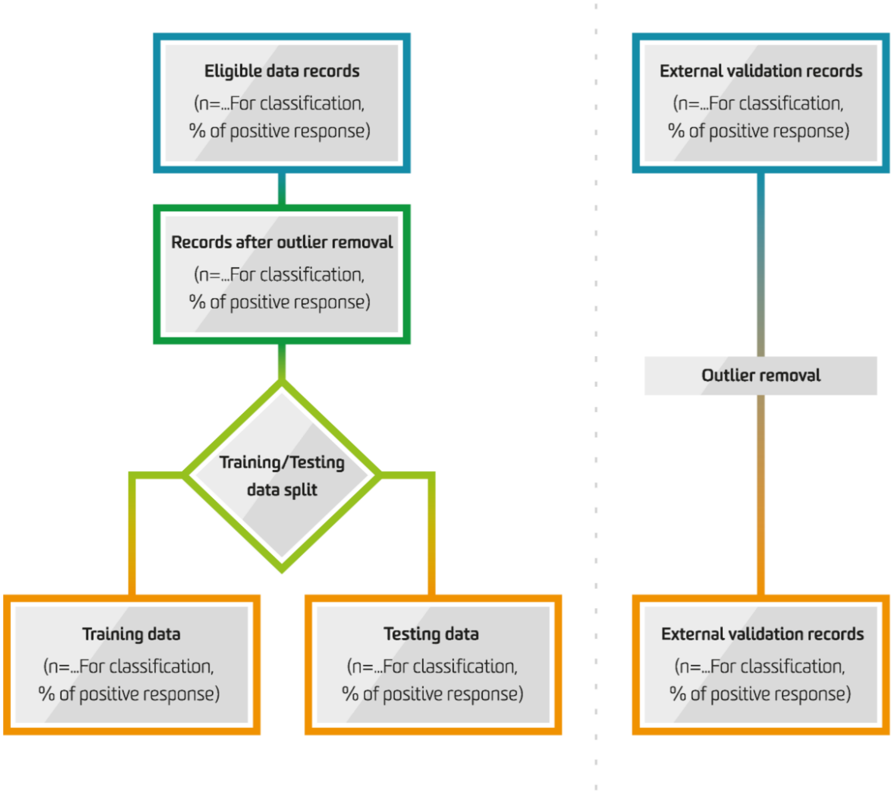

Eligible data records
(n=...For classification, % of positive response)
Records after outlier removal
(n=...For classification, % of positive response)
Training/Testing data split
Training data
(n=...For classification, % of positive response)
Testing data
(n=...For classification, % of positive response)
External validation records
(n=...For classification, % of positive response)
Outlier removal
External validation records
(n=...For classification, % of positive response)

Fig 1.13: Steps followed for training & testing a ML model.

(e) **Conversion of Android samples into Images**

After the implementation of multiple Machine Learning algorithms on the dataset created by extracting the permissions of Android samples, for obtaining more realistic and correct results, we used complex deep learning algorithms especially Convolutional Neural Network. The CNN works best for images as input. Thus, we have converted the Android samples into images. The format of images are .jpg and are the greyscale images of Android samples. For this, we used a python script using image libraries that converts the Android samples into images.

(f) **Normalisation of Images**

As the Android samples are converted to greyscale images, the images formed could be of various sizes and dimensions depending on the respective size of each application and thus the images need to be normalised for the input images to be in the same range. The images were later renamed using a python script which identifies the unique MD5 hash which acts as a fingerprint of the Android samples and distinguishes the sample from other samples and helps remove duplicacy.

(g) **Using Convolutional Neural Network**

After the creation of image dataset, a python script is run to classify the images into benign or malicious. During this process, the CNN model uses splitting of the dataset and splits the dataset for both training and testing of the model. This helps to determine the accuracy, precision and other parameters and give the result as to how effective the model is for the created dataset.

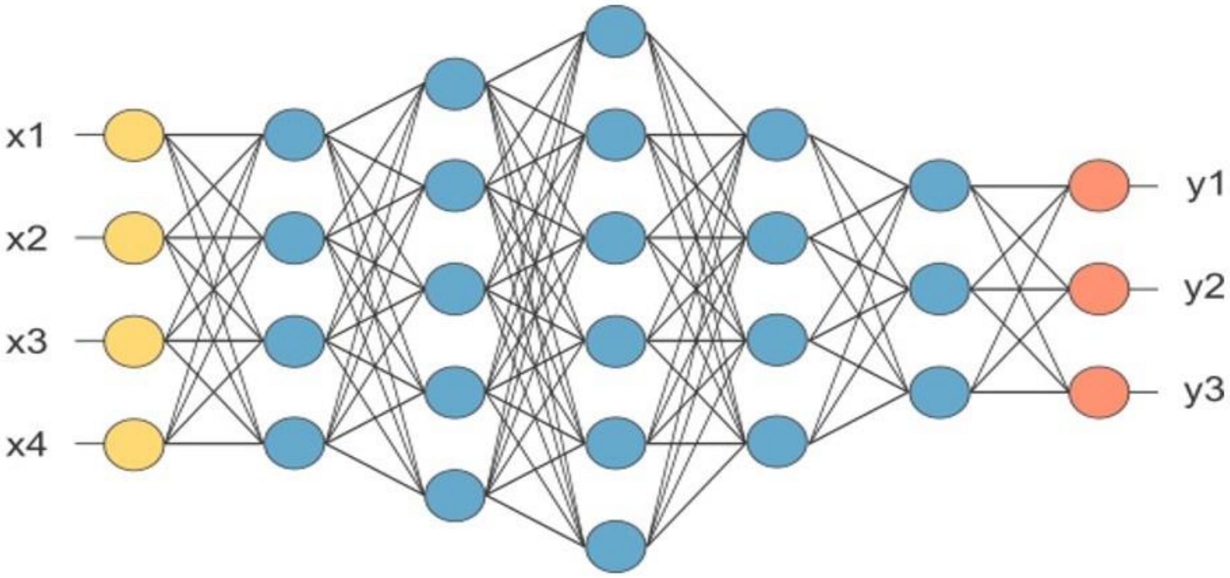


Fig 1.14: Convolutional Neural Network diagram

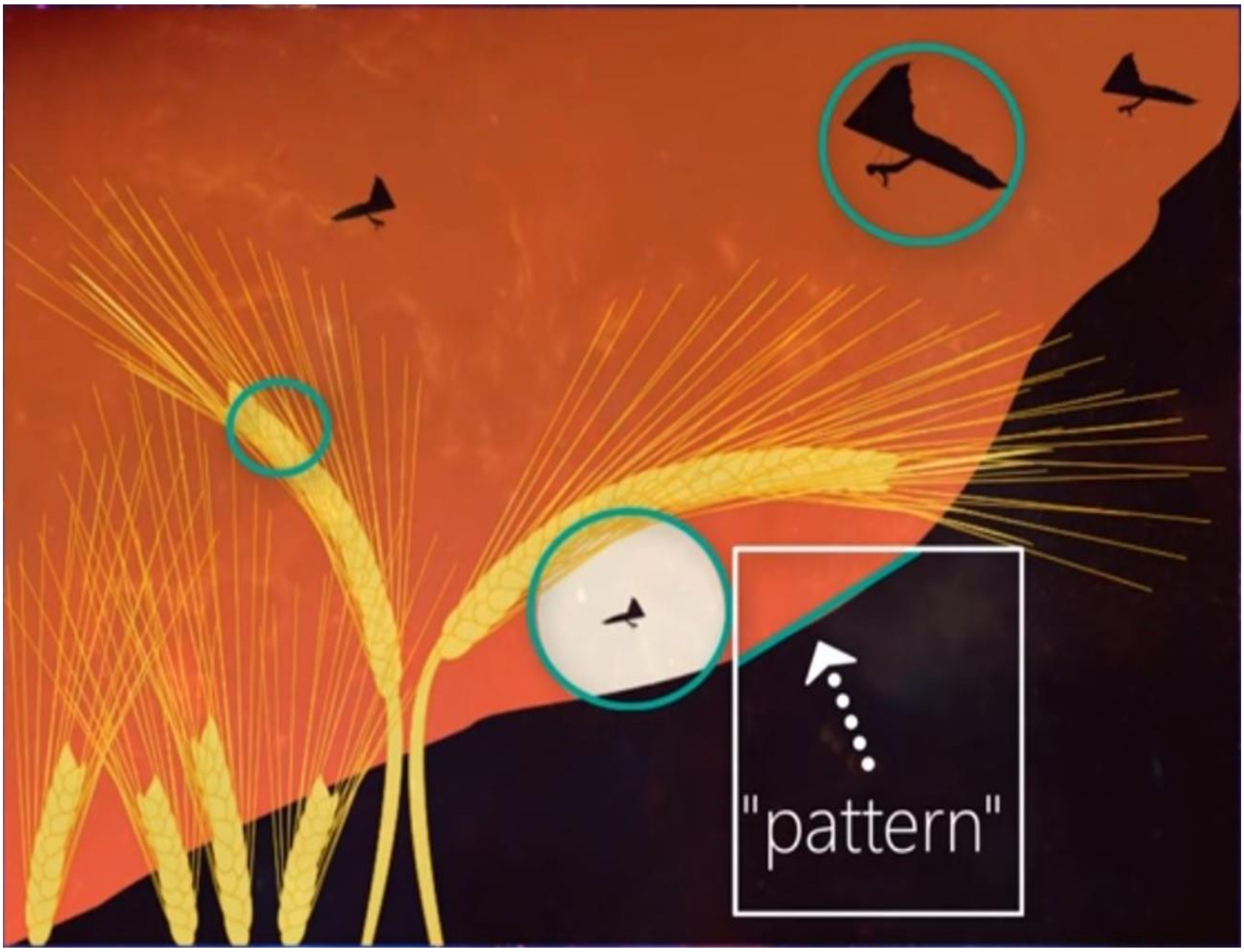


Fig 1.15: Convolutional Neural Network after first layer

In the process of convolutional neural network, the input image passes through multiple layers and each layer has its own filters and activation function of their respective perceptrons. In the first layer of convolutional neural network, the model only identifies the basic shapes such as circles, lines or other basic shapes. More and more complex filters and activation functions are to be used in the futher layers to identify more detailed features and extract useful information from the images. In the image above, the filter detecting the edges are called as edge detector.

As the layers increase, that is, as the model go deeper, we are able to use more complex filters and activation functions to extract more detailed information. In the next image, the example is shown after the n=4 layers and we can obtain multiple information from multiple images. The information obtained from this layers gives out dog faces, bird legs etc. Thus, as we go deeper and deeper, we can extract more information from the images.

On the other hand, as we go deeper and increase the number of layers in the model, the computation speed and the runtime increases. This pattern detection makes CNN so useful for image analysis.

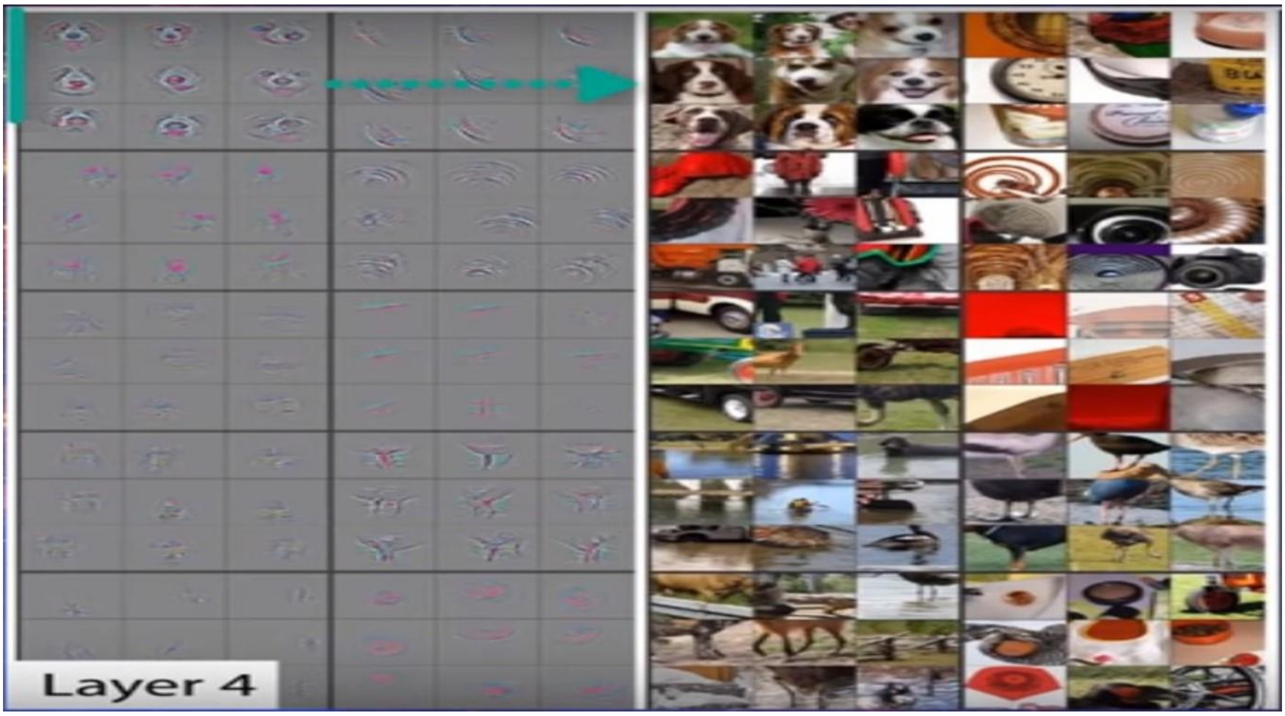


Fig 1.16: Convolutional Neural Network after n layers

The following image shows the example of the convolutional neural network used for handwriting detection. It is the most widely used and popular example of CNN and the model used has four convolutional layers with different and complex filters. An example of the filter being used in a convolutional layer is shown by a 3*3 matrix having random values. After all the layers, the result is convolved and we get the desired output or information.

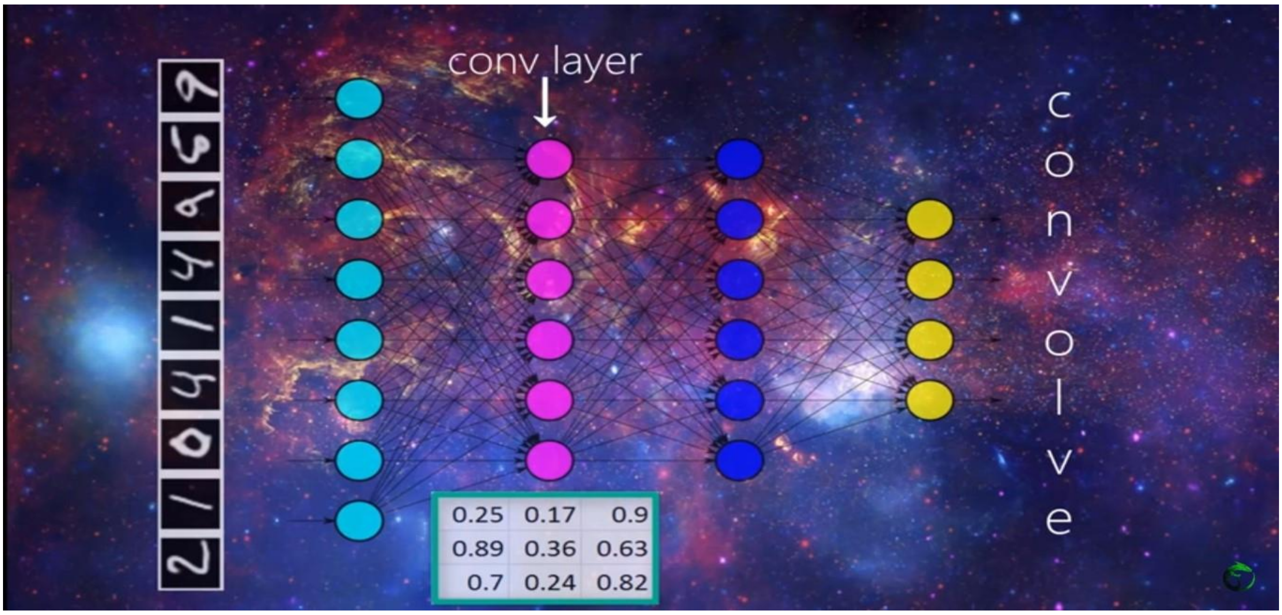
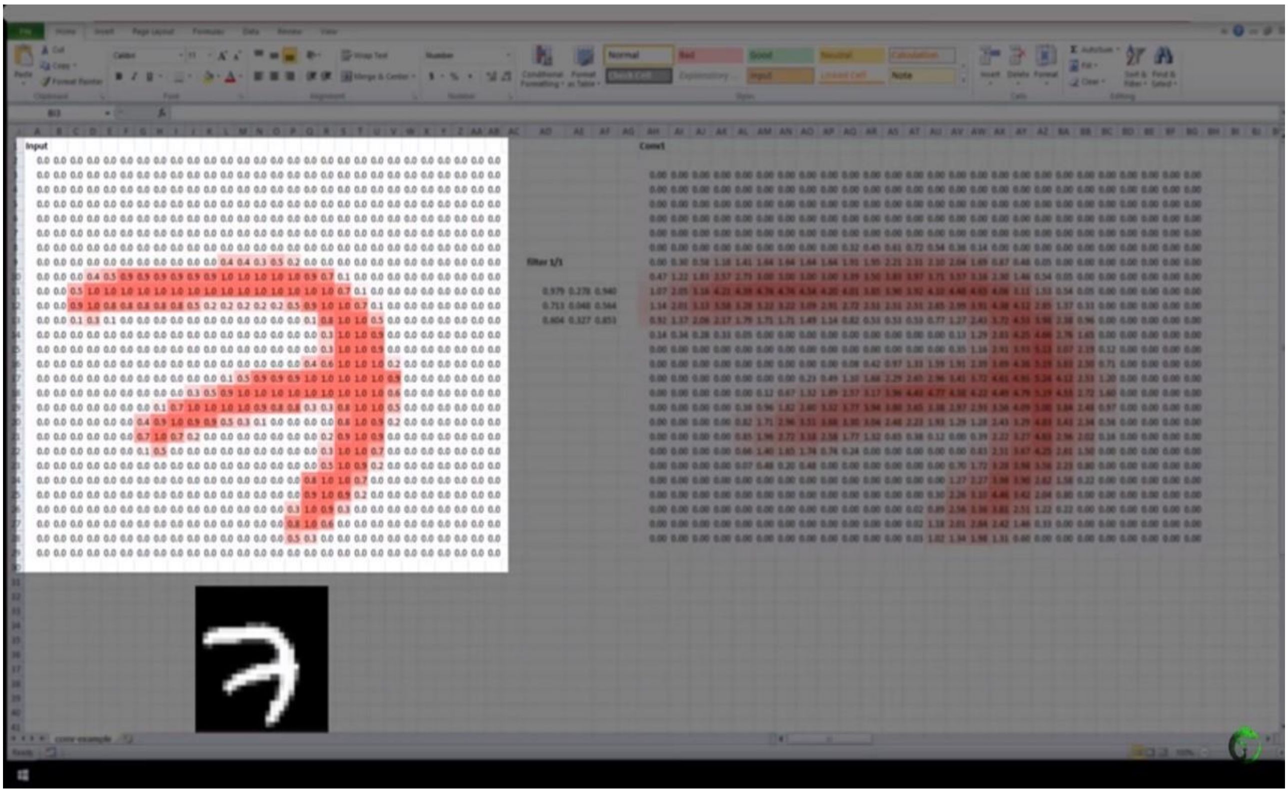


Fig 1.17: Handwriting detection example of CNN

In the next image, the working of convolutional neural network for the identification of handwriting is explained. The basis of CNN is that it receives an input and transforms the input in a way and then outputs the transform input to the next layer. With each convolutional layer, we need to specify the number of filters the layers should have. The filter is a matrix where the number of rows and number of columns are decided by the user and the values are initialized with random numbers. The matrix representation of an image of a 7 as given below is taken as an input by the convolutional layer. The values from the matrix are the individual pilexs from the image.



Input

0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.4	0.4	0.3	0.5	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.4	0.5	0.9	0.9	0.9	0.9	0.9	0.9	1.0	1.0	1.0	1.0	1.0	0.9	0.7	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.5	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.7	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.9	1.0	0.8	0.8	0.8	0.8	0.8	0.5	0.2	0.2	0.2	0.2	0.2	0.5	0.9	1.0	1.0	0.7	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.1	0.3	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.8	1.0	1.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3	1.0	1.0	0.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3	1.0	1.0	0.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.4	0.6	1.0	1.0	1.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.5	0.9	0.9	0.5	1.0	1.0	1.0	1.0	1.0	0.9	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.5	0.9	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.6	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.7	1.0	1.0	1.0	1.0	0.9	0.8	0.8	0.3	0.3	0.8	1.0	1.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.4	0.9	1.0	0.9	0.9	0.5	0.3	0.1	0.0	0.0	0.0	0.0	0.8	1.0	0.9	0.2	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.7	1.0	0.7	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.9	1.0	0.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3	1.0	1.0	0.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	1.0	0.9	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.8	1.0	1.0	0.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.9	1.0	0.9	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3	1.0	0.9	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.8	1.0	0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

filter 1/1

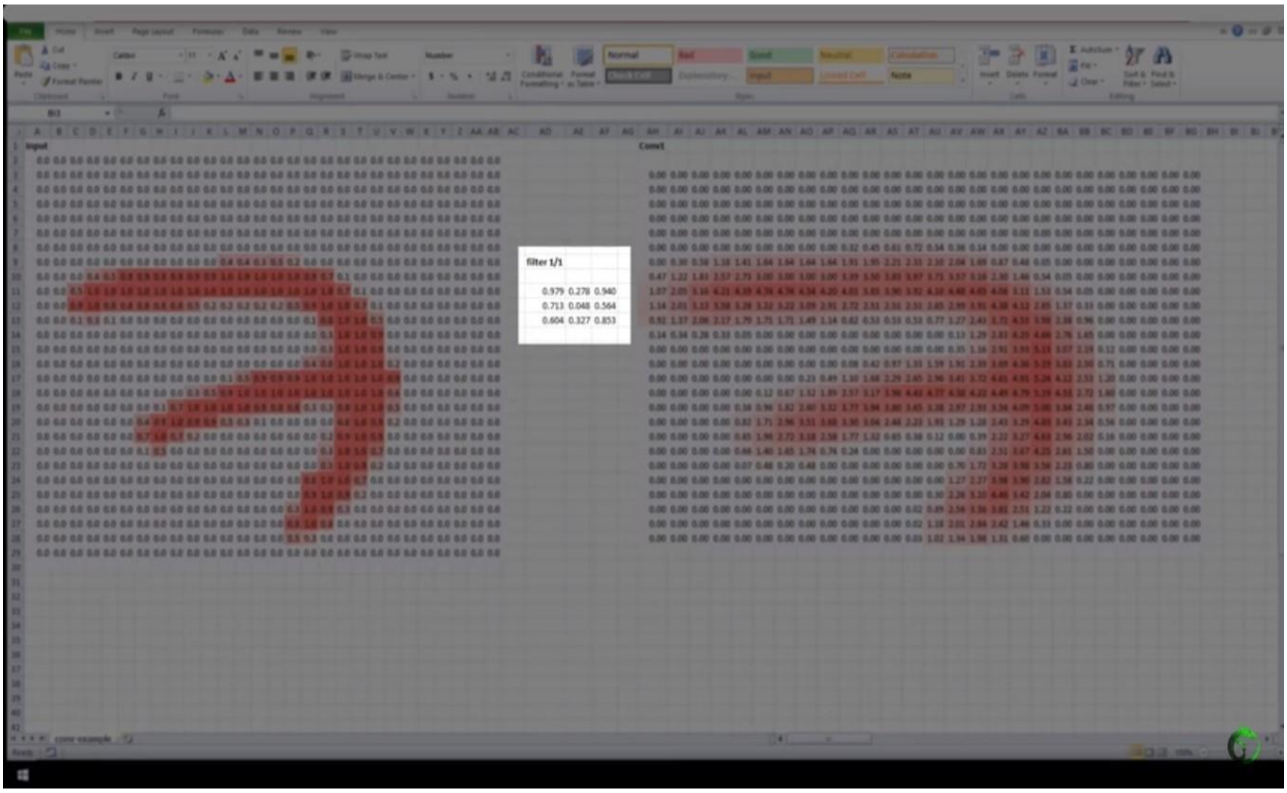
0.979	0.278	0.940
0.713	0.048	0.564
0.604	0.327	0.853

Conv1

0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.32	0.45	0.61	0.72	0.54	0.36	0.14	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.30	0.58	1.18	1.41	1.64	1.64	1.64	1.64	1.91	1.95	2.21	2.31	2.10	2.04	1.69	0.87	0.48	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.47	1.22	1.83	2.57	2.73	3.00	3.00	3.00	3.00	3.39	3.50	3.83	3.97	3.71	3.57	3.16	2.30	1.46	0.54	0.05	0.00	0.00	0.00	0.00	0.00	0.00
1.07	2.05	3.16	4.21	4.39	4.76	4.76	4.54	4.20	4.01	3.85	3.90	3.92	4.10	4.48	4.65	4.06	3.11	1.53	0.54	0.05	0.00	0.00	0.00	0.00	0.00
1.14	2.01	3.13	3.58	3.28	3.22	3.22	3.09	2.91	2.72	2.51	2.51	2.51	2.65	2.95	3.91	4.38	4.12	2.85	1.37	0.33	0.00	0.00	0.00	0.00	0.00
0.92	1.37	2.06	2.17	1.79	1.71	1.71	1.49	1.14	0.82	0.53	0.53	0.53	0.77	1.27	2.43	3.72	4.53	3.98	2.38	0.96	0.00	0.00	0.00	0.00	0.00
0.14	0.34	0.28	0.33	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.13	1.29	2.83	4.25	4.66	2.76	1.65	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.35	1.16	2.91	3.93	5.13	3.07	2.19	0.12	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.08	0.42	0.97	1.33	1.59	1.91	2.39	3.69	4.36	5.19	3.83	2.30	0.71	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.23	0.49	1.10	1.68	2.29	2.65	2.96	3.41	3.72	4.61	4.91	5.24	4.12	2.53	1.20	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.12	0.67	1.32	1.89	2.57	3.17	3.96	4.40	4.77	4.38	4.22	4.49	4.79	5.29	4.51	2.72	1.60	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.38	0.96	1.82	2.60	3.52	3.77	3.94	3.80	3.65	3.38	2.97	2.93	3.56	4.09	5.00	3.84	2.48	0.97	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.82	1.71	2.96	3.51	3.68	3.30	3.04	2.48	2.23	1.93	1.29	1.28	2.43	3.29	4.83	3.43	2.34	0.56	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.85	1.96	2.72	3.18	2.58	1.77	1.32	0.65	0.38	0.12	0.00	0.39	2.22	3.27	4.63	2.96	2.02	0.16	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.66	1.40	1.65	1.74	0.76	0.24	0.00	0.00	0.00	0.00	0.00	0.83	2.31	3.67	4.25	2.61	1.50	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.07	0.48	0.20	0.48	0.00	0.00	0.00	0.00	0.00	0.00	0.70	1.72	3.28	3.98	3.56	2.23	0.80	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.27	2.27	3.86	3.90	2.62	1.58	0.22	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.30	2.26	3.33	4.46	3.42	2.04	0.80	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.89	2.56	3.36	3.81	2.51	1.22	0.22	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	1.18	2.01	2.84	2.42	1.46	0.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	1.02	1.34	1.98	1.31	0.60	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Fig 1.17: Matrix Representation of an image of 7 as input.

This is the input and this input will be passed to convolutional layer. In this layer, it is specified to have only one filter. This filter further convolves across each 3 * 3 blocks of pixels form the input. The 3*3 filter of randomly initialized values are given in the next image. When the filter convolves the first 3*3 blocks of pixels of the input matrix, the dot product of the 3*3 blocks of pixels of filter with this 3*3 blocks of pixels of input matrix will be computed and stored it a new matrix as a single value and this output further transforms as an input for further layers.



Input

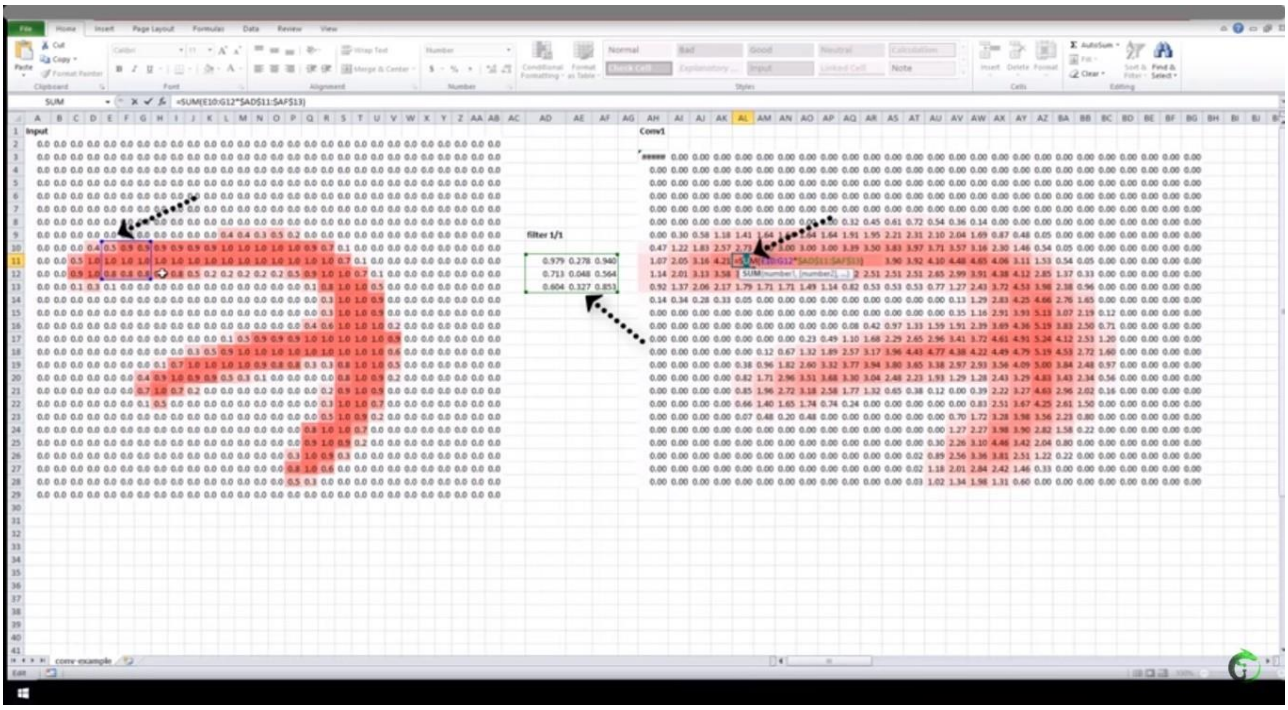
Conv1

filter 1/1

0.979	0.278	0.940
0.713	0.048	0.564
0.604	0.327	0.853

Fig 1.18: Matrix Representation of 3*3 blocks of pixels of filter.

Now, we slide to the next 3*3 block and take the dot product of the next 3*3 blocks of pixels with the 3*3 blocks of pixels of filter and the output is computed and stored as the next value in the output matrix. This process goes on for each 3*3 blocks of pixels of input and the output is stored and passed to the next layer where the same process is again repeated but with a different filter and activation function.



SUM

=SUM(E10:G12*$AD$11:$AF$13)

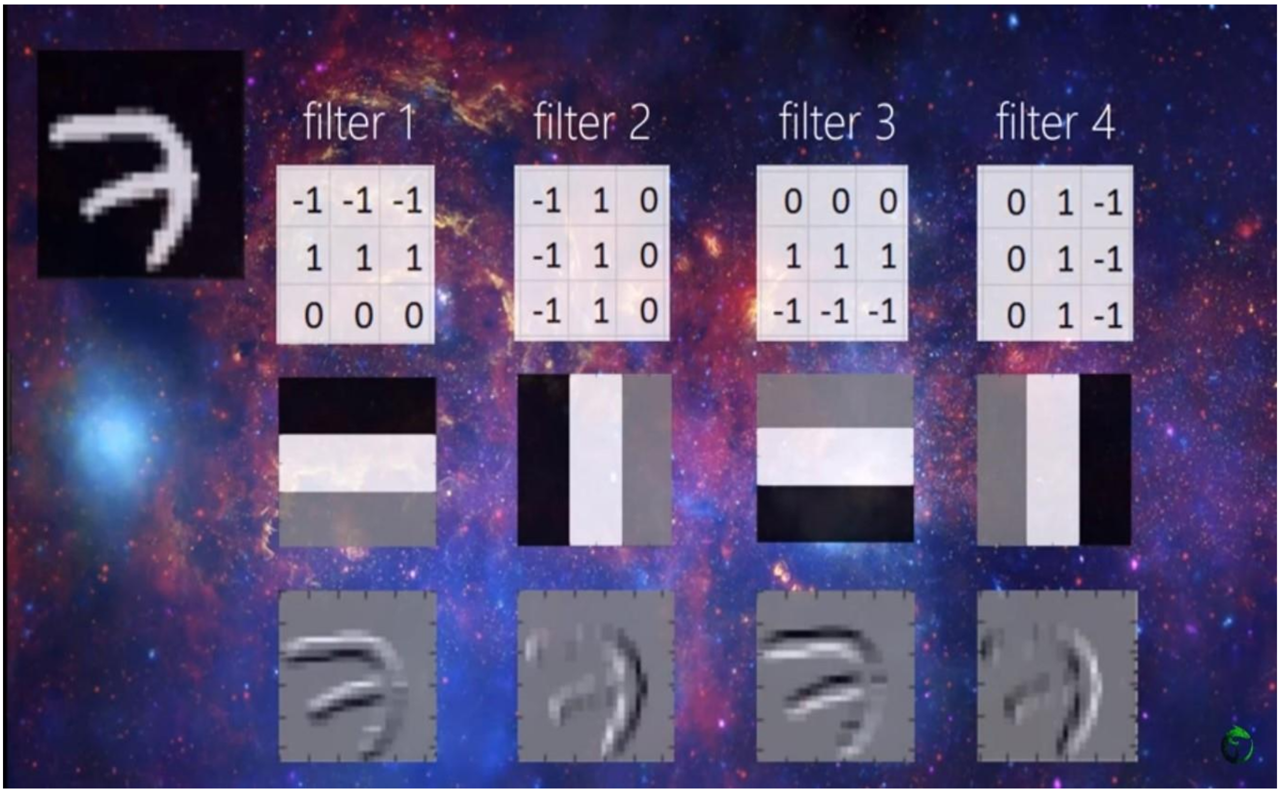
Input

filter 1/1

0.979	0.278	0.940
0.713	0.048	0.564
0.604	0.327	0.851

Conv1

Fig 1.19: Computation for filter and input and stored in output matrix.



filter 1

-1	-1	-1
1	1	1
0	0	0

filter 2

-1	1	0
-1	1	0
-1	1	0

filter 3

0	0	0
1	1	1
-1	-1	-1

filter 4

0	1	-1
0	1	-1
0	1	-1

Fig 1.20: Computation for filter and input and stored in output matrix.

In the above image, the input image of a 7 is convoluted with four different filters and the values of filters and given above and these values can be represented visually as the images shown below the matrixes. Here in the images, -1 correspond to the black, 1 corresponds to the white and 0 corresponds to the grey. After the convolution of the input image of 7 with these filters, we get the images as shown as output. Here, we can see that all the filters detect the edges and the bright part corresponds to the white in this example. These filters are really basic and mostly detects edges. More complex filters would be located deeper in the network and gradually be able to detect more sophisticated information.

(1.5) ORGANIZATION OF REPORT

In Chapter **2**, we have discussed the literature review, which consists of all the terminologies, importance and the working of the algorithms and its types. We have described various algorithms and its advantages and disadvantages to identify the best suited algorithm for our project.

In Chapter **3**, we have developed a system design and listed all the system requirements to run these algorithms and the environment on which these algorithms were tested to clearly analyse the data to get better results.

In Chapter **4**, we have discussed about all the algorithms used and the mathematics or the formulation behind these algorithms for better understanding and discussed about how all these algorithms were used.

In Chapter **5**, we have discussed about the testing of the model and observations of the results.

In Chapter **6**, we have discussed about the results and done analysis on the basis of various parameters to obtain the best suited algorithm for our dataset.

In Chapter **7**, we have thereby put an end and concluded the report and listed the conclusion learning of the project. Also we have discussed about the future scope and the advancements that could be carried out in the project.

Chapter-2

LITERATURE REVIEW

(2.1) Terminologies

(a). Machine Learning- Machine learning algorithms are a type of algorithms that are a branch of artificial intelligence and that makes the system or the software application to be smart enough to be able to more accurate without being explicitly programmed and can predict outcomes. The main idea behind these type of algorithms is that it receives input data in the form of text or images and the system or the model is trained with the statistical inputs to identify or predict the output and even updated the outputs as new data becomes available. It requires the algorithm to search through the dataset and look for patterns or similarities and manipulating or adjusting the system accordingly.

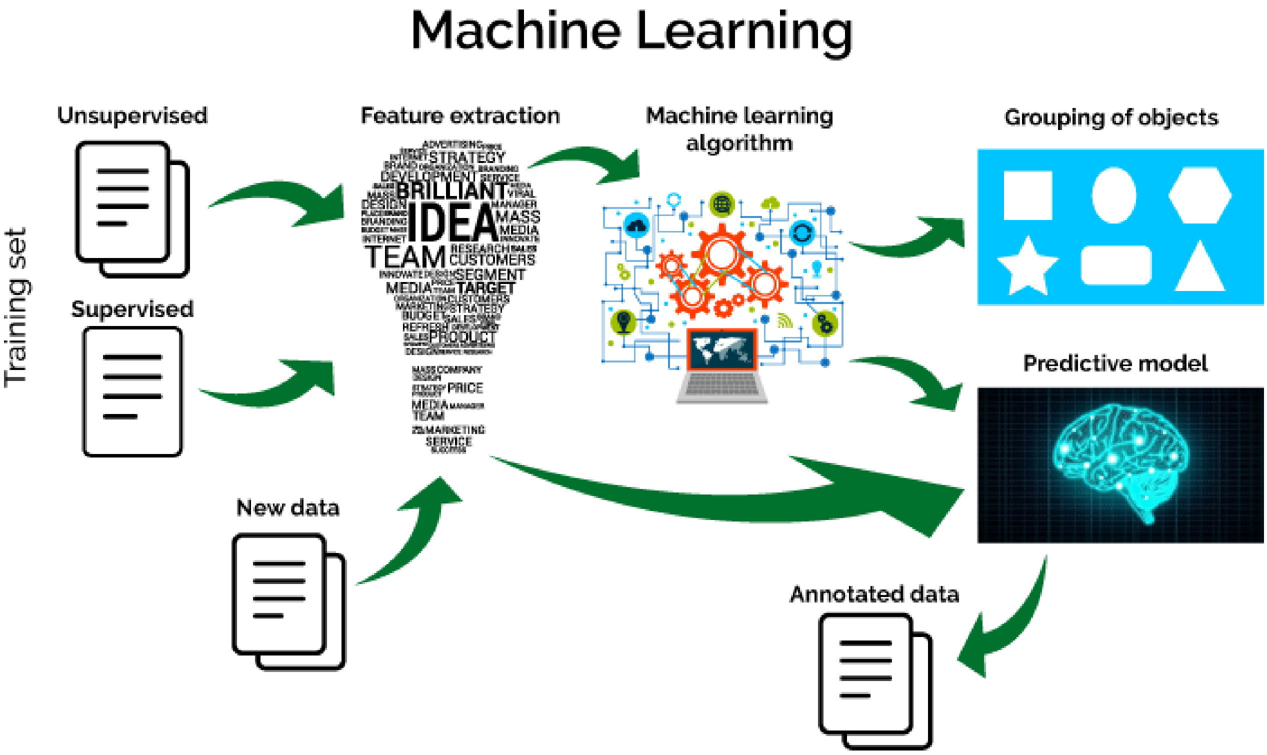
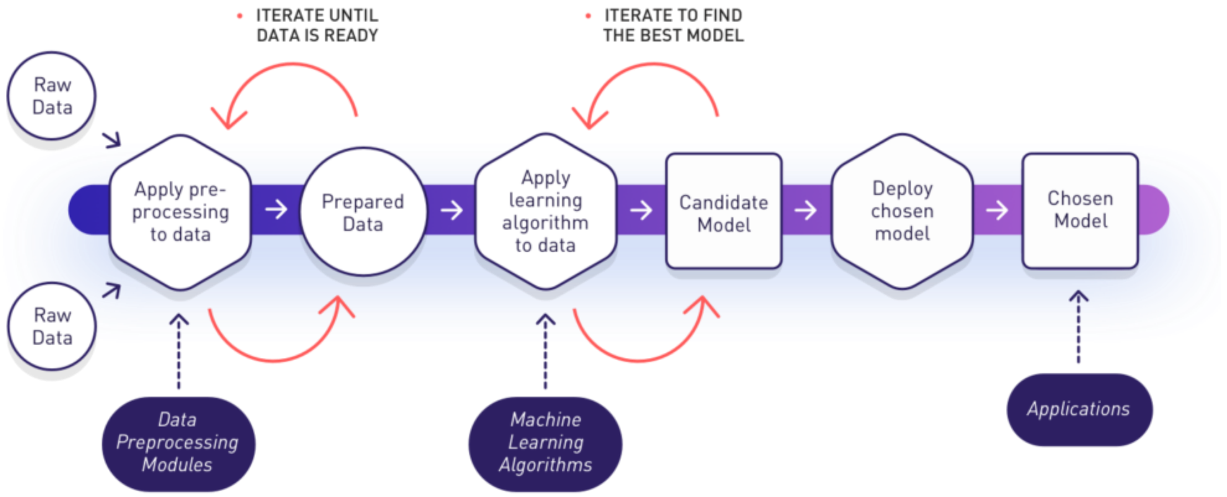


Fig 2.1: Introduction to ML

(b). How machine learning works

The process of machine learning starts with the collection of data or observations as the input dataset which can be in the form of images, text, tables etc. Further, many predefined machine learning algorithms are applied to the input data which either classify the data into groups or identifies patterns among the dataset to predict the output and give appropriate results. Machine learning algorithms are loosely classified into supervised and unsupervised learning algorithms.



ITERATE UNTIL DATA IS READY
ITERATE TO FIND THE BEST MODEL
Raw Data
Raw Data
Apply pre-processing to data
Prepared Data
Apply learning algorithm to data
Candidate Model
Deploy chosen model
Chosen Model
Data Preprocessing Modules
Machine Learning Algorithms
Applications

Fig 2.2: ML algorithm workflow

(c). Types of Machine Learning

1. Supervised Machine learning- This type of algorithms work for a dataset which is already being trained by previous outputs and outcomes of the past using labelled data to predict the outcome of the new data. In this case the known dataset is analysed, the algorithm then produces an inferred function which help in prediction of the output values of new data. It can also analyse the data and the outcome and compare with the previously stored data to find errors and to be able to modify and train the model accordingly.

2. Unsupervised Machine Learning- This type of different from supervised machine learning algorithms as this algorithms are used when the model is not trained before neither it is classified nor it is labelled. Unsupervised learning algorithms make the system to infer a hidden structure or pattern in the unlabelled dataset and predict possible results with the use of such patterns while removing the outliers.

3. Semi-Supervised Machine Learning- The semi-supervised machine learning algorithms are algorithms which uses merits of both supervised and unsupervised machine learning algorithms for training the dataset and thus produces much productive and powerful classifiers. In these type of algorithms, the model uses both labelled and unlabelled data for the training and it mostly requires a small amount of labelled data and a relatively large number of unlabelled data which are used simultaneously to train the model. This is used for enhancing the accuracy and the prediction abilities of the model and thus often used in the case of data which requires both skilled and relevant sources for training and learning from it.

4. Reinforcement Machine Learning- The main idea of reinforcement learning is that it is reward based training in which the model interacts with the environment by doing actions and discovering errors or rewards. The most relevant characteristics of reinforcement learning are the trial and error and delayed reward. In this case the model learns from its mistakes or errors and the model is made to interact with the machines to automatically determine the outcome and the ideal behaviour to enhance the working and for performance maximization.

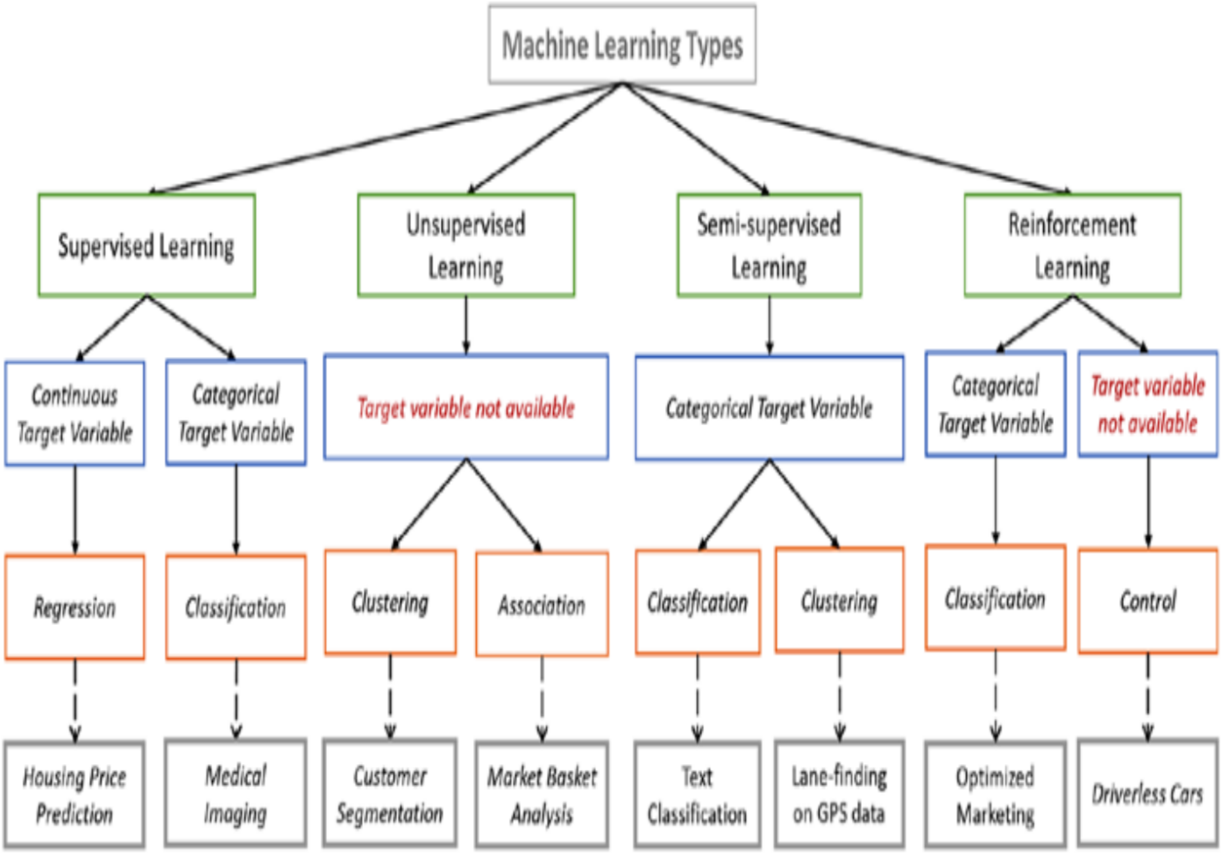


Fig 2.3: Types of machine learning

(d). Machine Learning Algorithms-

This section contains of various machine learning algorithms which are to be used in the project and discussed-

- **Supervised learning-**
    - **Nearest Neighbors-** KNN or K-nearest neighbors is a type of algorithm which can be used both for regression and classification problems but is mostly used in classification problems. This algorithm is easy in interpretation and requires very low calculation time and thus is a widely used ML algorithm. The K in this algorithm is the number of neighbors which are defined by the user. In this algorithm we use the Euclidean distance to measure the K nearset neighbors of the data point and predict the output according to its neighbors.

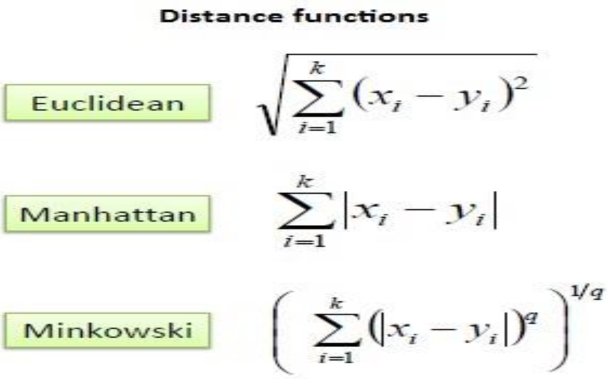

Distance functions
Euclidean
$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$
Manhattan
$\sum_{i=1}^{k}|x_i - y_i|$
Minkowski
$\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$

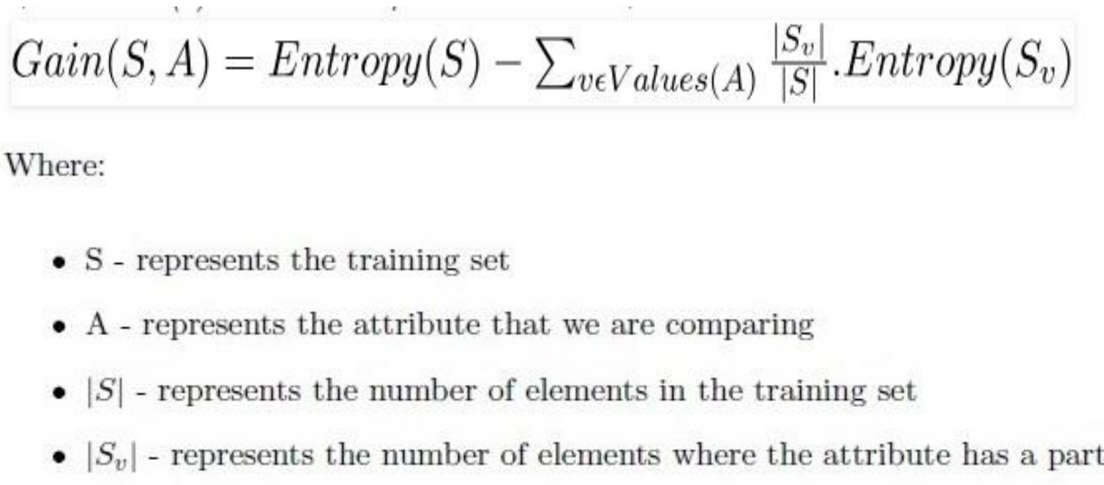
Fig 2.4: Formulae used for calculating nearset distance.

- **Naïve Bayes-** Naive Bayes theorem is a type of classification algorithm which can be used for both binary and multi class classification problems. This theorem is called so because it has its roots of Bayes theorem. Naïve Bayes is often represented by probabilities. In this model the data is stored as probabilities for a learned model.

  Formulation- **P(h|d) = (P(d|h)*P(h))/ P(d)**

- **Decision Trees-** Decision tree algorithm is a type of supervised learning algorithm in which a data structure is used to solve a problem. In this case the leaf node is referred to as the class label and the internal nodes of the tree represent the attributes. They are able to solve the problems of both classification and regression. Initially, we consider the whole dataset as the root and categorical feature values are preferred and the continuous values are first made discrete values before using them to build the model. Then statistical methods are used for ordering the attributes as internal node or root.

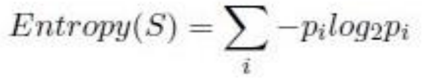
  Formulation-



$$Gain(S,A) = Entropy(S) - \sum_{v \epsilon Values(A)} \frac{|S_v|}{|S|}.Entropy(S_v)$$

Where:

- S - represents the training set
- A - represents the attribute that we are comparing
- $|S|$ - represents the number of elements in the training set
- $|S_v|$ - represents the number of elements where the attribute has a part



$$Entropy(S) = \sum_i -p_i log_2 p_i$$

- **Linear Regression-** Linear regression is an algorithm which uses the statistical concepts and models a relationship between the input and output numerical values. The model is represented by a linear equation which combines the input values of a specific set and predicts the output for a set of that input values.

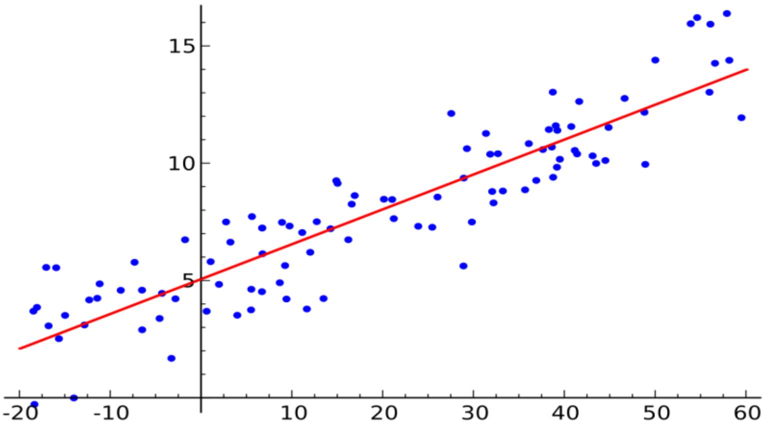


Fig 2.5: Pictorial representation of example of LR.

- **Support Vector Machines-** SVM is a supervised machine learning algorithm which is commonly used for both regression and classification problems. It is widely used in classification problems where each data item is plotted in n-dimensional space and n defines the features present and the value of each feature is the value of each coordinate. Further, a separate hyper plane is made to differentiate the two classes.
-

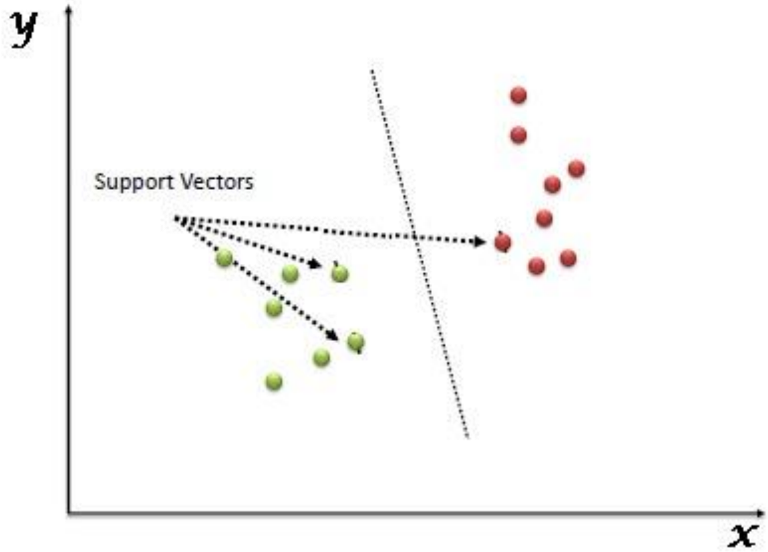
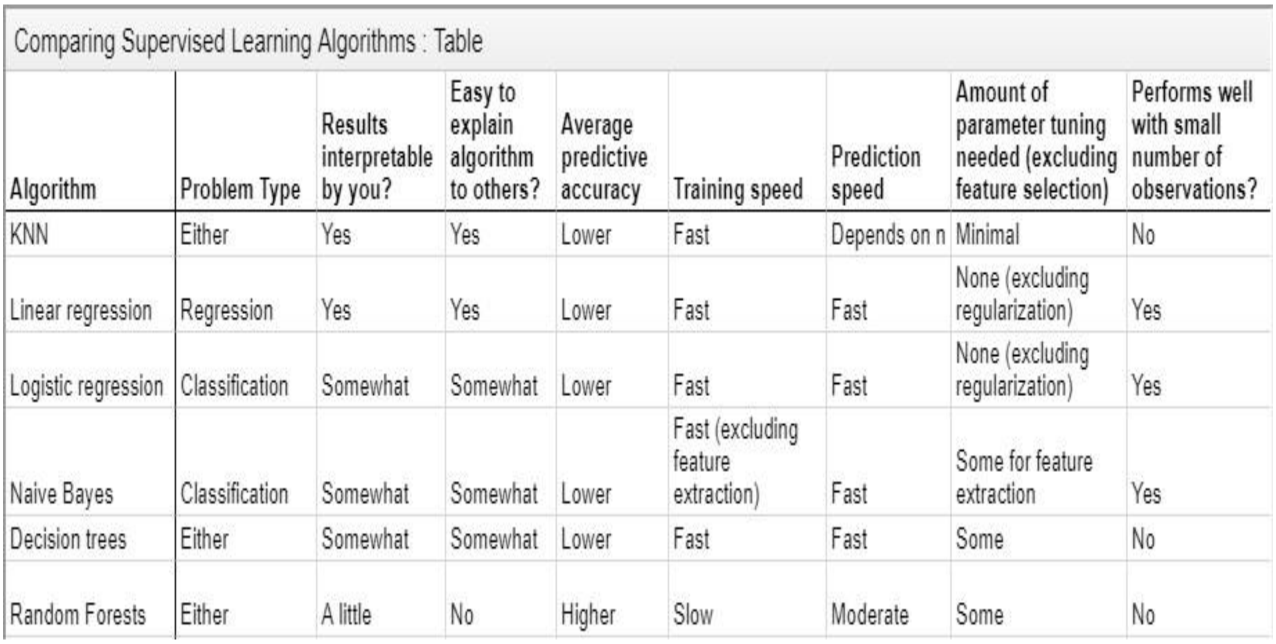


Fig 2.6: Pictorial representation of SVM.



Comparing Supervised Learning Algorithms : Table

Algorithm	Problem Type	Results interpretable by you?	Easy to explain algorithm to others?	Average predictive accuracy	Training speed	Prediction speed	Amount of parameter tuning needed (excluding feature selection)	Performs well with small number of observations?
KNN	Either	Yes	Yes	Lower	Fast	Depends on n	Minimal	No
Linear regression	Regression	Yes	Yes	Lower	Fast	Fast	None (excluding regularization)	Yes
Logistic regression	Classification	Somewhat	Somewhat	Lower	Fast	Fast	None (excluding regularization)	Yes
Naive Bayes	Classification	Somewhat	Somewhat	Lower	Fast (excluding feature extraction)	Fast	Some for feature extraction	Yes
Decision trees	Either	Somewhat	Somewhat	Lower	Fast	Fast	Some	No
Random Forests	Either	A little	No	Higher	Slow	Moderate	Some	No

Table 2.1: Difference between various ML Algorithms

(e). Interpretation of Performance Measures

There are various methods to evaluate the performance of the algorithms. One of these methods is to determine the area under the curve or the ROC curve and other parameters which are also known as Confusion Metrics. To evaluate the performance measure of the classification model for a dataset that gives the true values are known, the confusion matrix table is used.

	Predicted Class		
Actual Class		**Class = Yes**	**Class = No**
	Class = Yes	**True Positive**	**False Negative**
	Class = No	**False Positive**	**True Negative**

Table 2.2: Confusion Matrix

The table shown above is known as the confusion matrix and has four sections. The two section in the green are the True Positive and True Negative and these are the observations which are correctky predicted. The other two sections are in red because these values are wrongly predicted and thus needs to be minimized. These sections are false negative and

False Positive respectively and occurs when there is a contradiction between actual class and the predicted class.

- **True Positives (TP) -** These are the values which are correctly predicted and are positive values which can be described as the positive value of actual class and positive value of predicted class. It is denoted by TP.
- **True Negatives (TN)** - These are the values which are correctly predicted but negative values which refers to the negation of actual class and negation of predicted class. It is denoted by TN.
- **False Positives (FP) –** These are the values which are wrongly predicted but is true in real i.e. - when we have positive values of actual class but negation in predicted class.
- **False Negative (FN) –** These are the values which are wrongly predicted and negative in actual class.

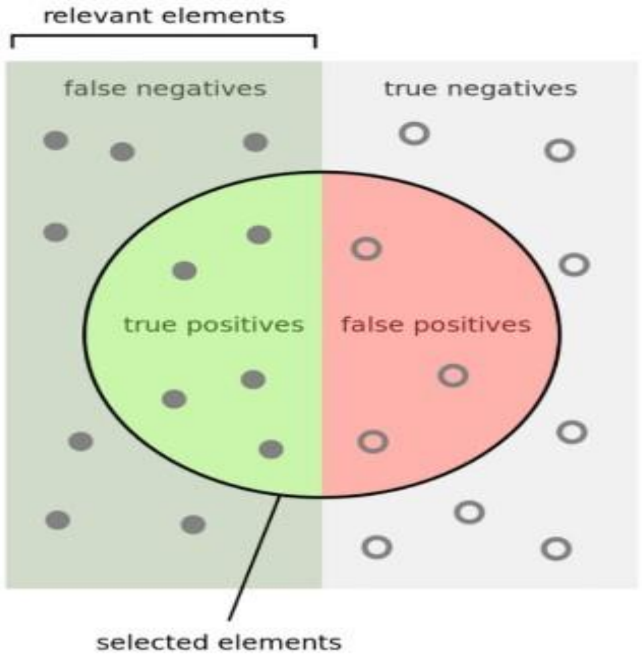


Fig 2.7: Pictorial representation of confusion matrix

Further, we look into more parameters of performance which are accuracy, precision, Recall and F1 score.

- **Accuracy** – Accuracy is the most natural execution measure and it is essentially a proportion of effectively anticipated perception to the aggregate perceptions. One may imagine that, on the off chance that we have high exactness, our model is ideal. Truly, exactness is an extraordinary measure yet just when you have symmetric datasets where estimations of false positive and false negatives are relatively same. In this manner, you need to take a gander at different parameters to assess the execution of your model. For our model, we have 0.986839 which implies our model is approx. 98% precise.

  Accuracy = TP+TN/TP+FP+FN+TN

- **Precision -** Precision is the proportion of accurately anticipated positive perceptions to the aggregate anticipated positive perceptions. The inquiry that this measurement answer is of all travellers that named as endure, what number of really endure? High precision identifies with the low false positive rate.

  Precision = TP/TP+FP

- **Recall (Sensitivity)** - Recall is the proportion of effectively anticipated positive perceptions to the all perceptions in genuine class - yes. The inquiry review answers is: Of the considerable number of travelers that really endure, what number of did we mark?

  Recall = TP/TP+FN

- **F1 Score -** F1 Score is the weighted normal of Precision and Recall. Consequently, this score considers both false positives and false negatives. Instinctively it isn't as straightforward as exactness, yet F1 is normally more valuable than precision, particularly on the off chance that you have an uneven class conveyance. Precision works best if false positives and false negatives have comparable expense. On the off chance that the expense of false positives and false negatives are altogether different, it's smarter to take a gander at both Precision and Recall.

  F1 Score = 2*(Recall * Precision) / (Recall + Precision)

(f) **Deep Learning-**

Deep learning is a more complex and intelligent sub category of machine learning which has its algorithms inspired by the functioning and structure of the human brain broadly known as Artificial neural network. In addition to this, it also refers to the collection of techniques which are used for learning in neural network with multiple layers. Artificial Neural Network or ANN are the type of neural network model which takes its inspiration and works on the basic idea of the nervous systems and the processing of information in human brain to learn from data. Here, the learning mechanisms can be either supervised, semi-supervised or unsupervised. Deep learning has been proven successful in various fields and resulted in more realistic learning of machines. Its uses ranges from the field of drug design to the traffic prediction and also for the object recognition. A deep neural network is different from a neural network because of the number of layers. While the implementation of deep learning, we encountered two main problems such as 1) the computational power needed for the process of training the model was higher than that of the system available and thus requires more time for computation. 2) Another problem that encountered during the implementation is the gradient vanishing problem, that is, in a neural network that has activation functions such as the hyperbolic tangent or the sigmoid and the gradient range is (-1,1) or [0,1), the backpropagation is usually computed by chain rule, multiplying k to this small numbers from the the output layer through a k-layer network, which means that the gradient decreased exponentially with k. Resulting of this is that the front layers of the model trains slowly than the other layers.

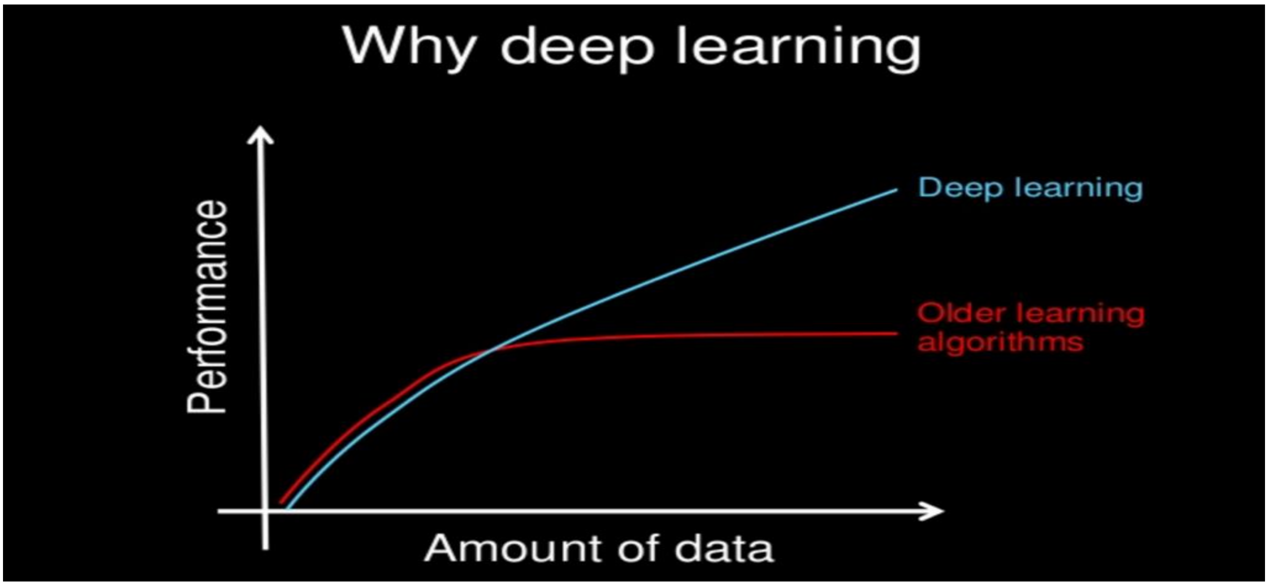


Fig 2.8: Pictorial representation of Deep learning Vs other learning algorithms

(g) **Perceptrons**

The most basic and the early supervised learning algorithm is the perceptron and it is the smallest building block of the network that is Artificial Neural Network. The working of the perceptrons is by taking multiple inputs (x_1,x_2……x_j) and production of a single output (y). In addition to this, weighted inputs were also considered to help determine the importance of respective inputs to the output. The resulting output is either 0 or 1 and it is only determined by checking if the weighted sum is greater than 0 or less than 0.

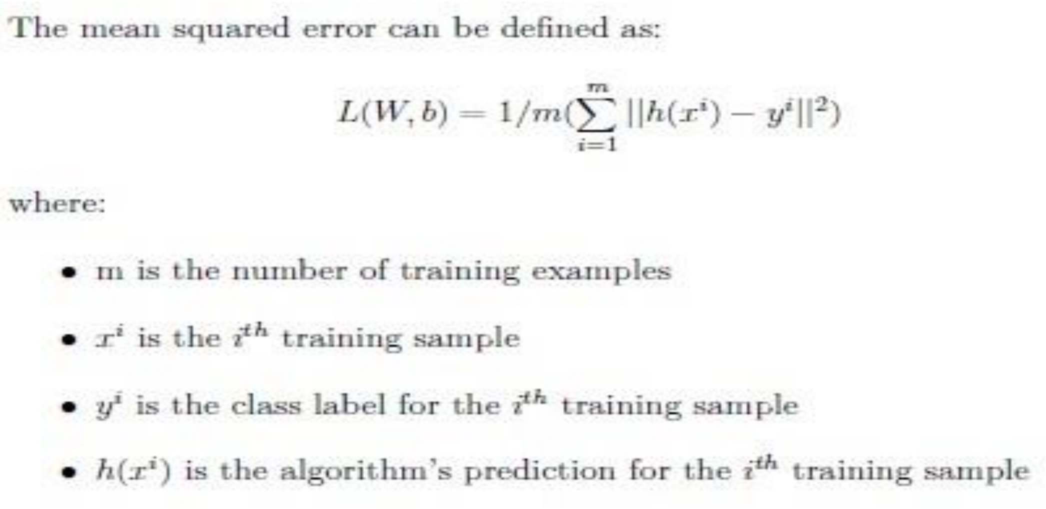
Weighted Sum- $\sum_j w_j * x_j + b$

$$\text{Output} = \left\{ \begin{array}{ll} 1: & \text{if } w * x + b > 0 \\ & \\ 0: & \text{if } w * x + b <= 0 \end{array} \right\}$$

However, as the perceptron only gives the output as 0 or 1, it makes it very difficult or nearly impossible to extend the working and functionalities of the model to be able to work on classification tasks having multiple categories. Furthermore, this problem can be resolved by having multiple perceptrons in a layers making sure that each perceptron in the layer obtains the same input and all the perceptrons are responsible for the output function. The Artificial Neural Network (ANN) is only perceptrons with more than one layers whereas the perceptron is nothing more than a ANN with single layer, which is often the output layer having only 1 neuron.

(h) **Loss Function**

The performance of the neural network is measured by a function which is called as the cost function or the loss function which helps in measuring the discrepancy between the prediction by the algorithms and the correct label if the prediction or the collective set of prediction is given along with the label or a set of labels. Among the various cost function available the most simple and the commonly used in neural networks is the mean squared error (MSE).



The mean squared error can be defined as:

$$L(W,b) = 1/m(\sum_{i=1}^{m} ||h(x^i) - y^i||^2)$$

where:

- m is the number of training examples
- x^i is the i^{th} training sample
- y^i is the class label for the i^{th} training sample
- $h(x^i)$ is the algorithm's prediction for the i^{th} training sample

The ultimate goal of training the neural networks is to minimize the cost / loss function and find the respective weights and biases that do so. For this procedure, we used an algorithm which called as gradient descent algorithm.

(2.2) Related Work

Due to the rise in the malware activities in the android community, more and more researchers are drawn towards the detection of malicious android samples and there have been many research efforts. Researchers from around the world follow different approaches to mitigate this issue. There have been many static as well as dynamic advancements in this field. Many static analysis approaches given in [2], [3], [4] follows basis on already known malware and compute applications through reverse engineering which help to decompile the packaged applications thereby making it easier to look for signatures or other heuristics written in the program code. Some follows different approaches like [5], [6], [7], which checks the usage of power by each application and reporting the user or developers about any anomalous consumption.

Many dynamic analysis approaches used in [8], [9], [10] keeps a check on the pattern of system calls. Others like [11], [12] have implemented the approach of universal signature based that are able to compare the application in question with many known malware or other heuristics.

Most of them like in [13], [14], [15], and [16] used the concept of machine learning with some concepts of data mining which made it relatively faster to detect malware.

In [9], a framework based on ML known as Crowdroid is used that is able to recognize malware on android smartphones like –trojan. It keeps a check on system calls and how many times each system call were issued. Similarly, in the case of Andromaly used in [17], which is an intrusion detection system based on machine learning. It keeps a check on the user's and well as smartphone's behaviour by observing certain parameters, spanning from sensor activities to CPU usage. This model uses 88 features which together describes the system behaviours along with rooting the system and the external use of Linux server. Further, these are pre-processed by feature selection mechanisms. A machine learning based clustering model is used in [18], this model analyses the static function calls from binaries to detect anomalies. The Symbian OS used this kind of technique. This framework uses a client, Remote Anomaly Detection System, which monitors and visualizes the component. In [19] Dini et al. presented a multi-level system called MADAM (Multi-Level Anomaly Detector for Android Malware) which is capable of extracting features at both the kernel and application levels using 13 features to describe system's behaviour and system calls. But this model was only targeted for rooted device. In [20] Portokalidis et al. presented a different approach based on VMM approach to detect malware in their design of Paranoid Android system where a full malware analysis can be done in the clouds using many replicas of mobile phone. A secure virtual environment is created for the mobile replicas to run and which makes it possible for approximately 105 replicas. Mirela et al [21] presented an approach based on neural network which were very efficient in detecting the fraud calls and imposter. The main drawback of this method is that it is a slow process and it classifies the samples into groups having same behaviour and thus will lead to lot of false positives. Furthermore, Jerry et al, [22] worked on the transmission of viruses through SMS messages or other interfaces like Bluetooth and infrared.

Chapter-3

SYSTEM DESIGN

(3.1) System Requirements

The algorithms that are being implemented in this project requires some generic system as it requires processing of algorithms.

- Windows 10 (64-bit)
- ANACONDA
- Python
- 4 GB RAM
- Intel(R) Core(TM) i3-3120M CPU @ 2.50 GHz

(3.2) Why Python

Python is a programming language with a large audience and it is very easy to understand and can be easily readable. Furthermore, python offers the collection of packages which makes the most intimidating algorithms or projects simpler. Python has libraries for almost every usable file i.e. - with working with images, working with text or working with audio files. Even when working with a new OS, python is very malleable. Python has a large community which makes it easier to seek for help and tips and tricks.

(3.3) Why ANACONDA

ANACONDA is widely popular as it provides all the libraries pre-installed and make the user free from hassle of the otherwise installing all libraries. It has around 100 packages which can be used for data science, machine learning or statistical analysis.

(3.4) SCIKIT LEARN

Scikit learn is a library in python usually used for machine learning and is capable of featuring various regression, classification and clustering algorithms.

(3.5) PANDAS

It is an open source python library which provides high performance. This library is easy to use and even provide data structure and data analysis tools. This library is widely used in all academic fields, commercial and industrial fields.

(3.6) KERAS

It is an open source python library that is generally used for neural network. It is often designed to run fast experimentation of many complex deep learning algorithms. It usually focuses on being more user-friendly, more modular and more extensible.

(3.7) Pillow

It is an open source python library that is used for imaging and also adds supports in the script for opening, also manipulating, further saving many different image file formats. It offers standard procedures for doing image manipulation. This includes per-pixel manipulations and masking and transparency handling.

(3.8) TENSERFLOW

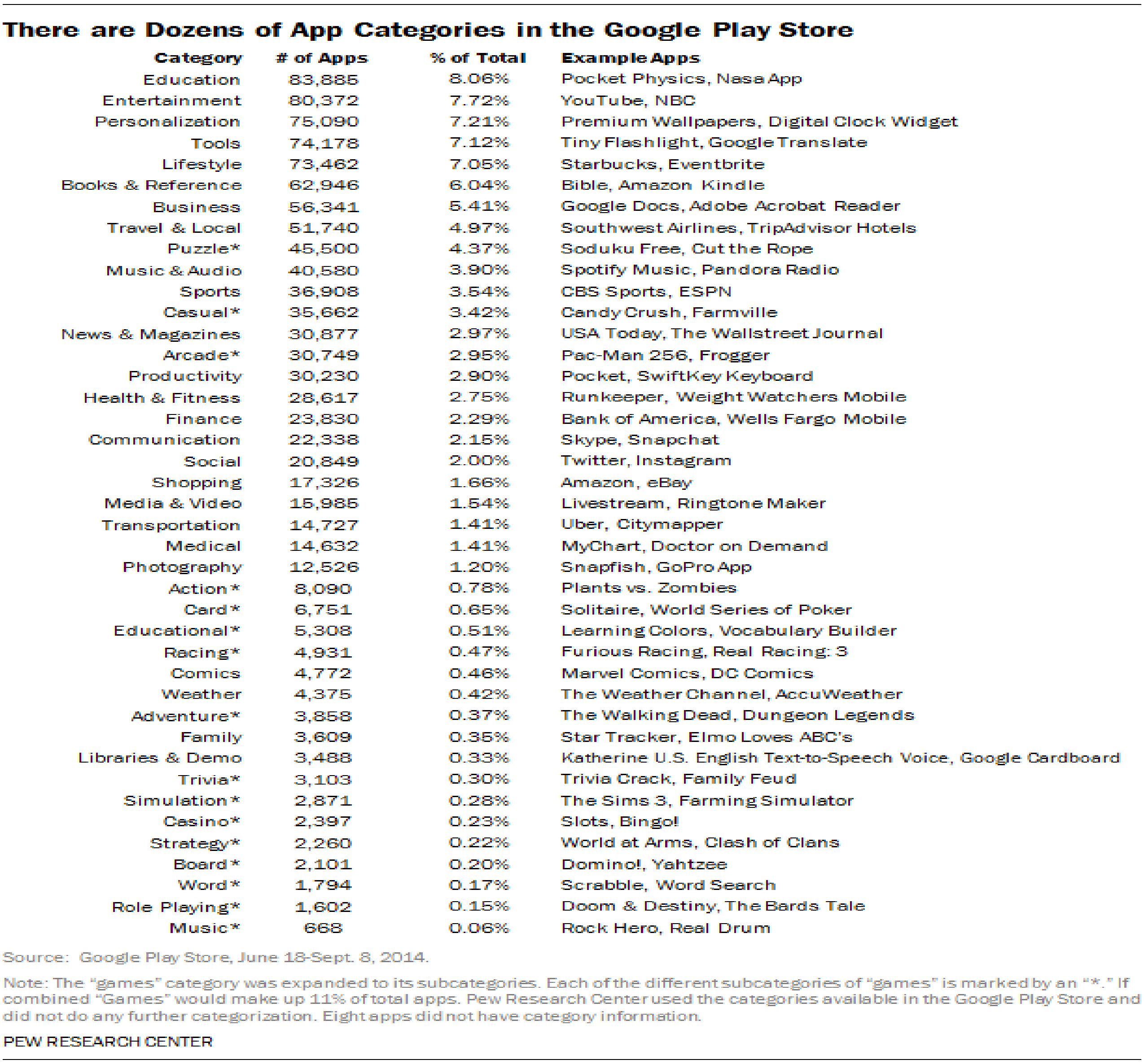
It is a free open source python library that is use for dataflow and differentiable programming which is used across a certain range of tasks. The tenserflow can be defined as a symbolic math library which can also be used for many machine learning applications for example neural networks.

Chapter-4

IMPLEMENTATION

In this chapter, we will discuss about all the phases of our project and the testing of our project.

1. In the first phase, we studied about the malware and the permissions and the different categories of permissions that an application seeks during its functioning. These permission categories were available on google play and google developer and gave clear understanding of how the working of the application is affected by each permission the system requires both for software and hardware access.



There are Dozens of App Categories in the Google Play Store

Category	# of Apps	% of Total	Example Apps
Education	83,885	8.06%	Pocket Physics, Nasa App
Entertainment	80,372	7.72%	YouTube, NBC
Personalization	75,090	7.21%	Premium Wallpapers, Digital Clock Widget
Tools	74,178	7.12%	Tiny Flashlight, Google Translate
Lifestyle	73,462	7.05%	Starbucks, Eventbrite
Books & Reference	62,946	6.04%	Bible, Amazon Kindle
Business	56,341	5.41%	Google Docs, Adobe Acrobat Reader
Travel & Local	51,740	4.97%	Southwest Airlines, TripAdvisor Hotels
Puzzle*	45,500	4.37%	Soduku Free, Cut the Rope
Music & Audio	40,580	3.90%	Spotify Music, Pandora Radio
Sports	36,908	3.54%	CBS Sports, ESPN
Casual*	35,662	3.42%	Candy Crush, Farmville
News & Magazines	30,877	2.97%	USA Today, The Wallstreet Journal
Arcade*	30,749	2.95%	Pac-Man 256, Frogger
Productivity	30,230	2.90%	Pocket, SwiftKey Keyboard
Health & Fitness	28,617	2.75%	Runkeeper, Weight Watchers Mobile
Finance	23,830	2.29%	Bank of America, Wells Fargo Mobile
Communication	22,338	2.15%	Skype, Snapchat
Social	20,849	2.00%	Twitter, Instagram
Shopping	17,326	1.66%	Amazon, eBay
Media & Video	15,985	1.54%	Livestream, Ringtone Maker
Transportation	14,727	1.41%	Uber, Citymapper
Medical	14,632	1.41%	MyChart, Doctor on Demand
Photography	12,526	1.20%	Snapfish, GoPro App
Action*	8,090	0.78%	Plants vs. Zombies
Card*	6,751	0.65%	Solitaire, World Series of Poker
Educational*	5,308	0.51%	Learning Colors, Vocabulary Builder
Racing*	4,931	0.47%	Furious Racing, Real Racing: 3
Comics	4,772	0.46%	Marvel Comics, DC Comics
Weather	4,375	0.42%	The Weather Channel, AccuWeather
Adventure*	3,858	0.37%	The Walking Dead, Dungeon Legends
Family	3,609	0.35%	Star Tracker, Elmo Loves ABC's
Libraries & Demo	3,488	0.33%	Katherine U.S. English Text-to-Speech Voice, Google Cardboard
Trivia*	3,103	0.30%	Trivia Crack, Family Feud
Simulation*	2,871	0.28%	The Sims 3, Farming Simulator
Casino*	2,397	0.23%	Slots, Bingo!
Strategy*	2,260	0.22%	World at Arms, Clash of Clans
Board*	2,101	0.20%	Domino!, Yahtzee
Word*	1,794	0.17%	Scrabble, Word Search
Role Playing*	1,602	0.15%	Doom & Destiny, The Bards Tale
Music*	668	0.06%	Rock Hero, Real Drum

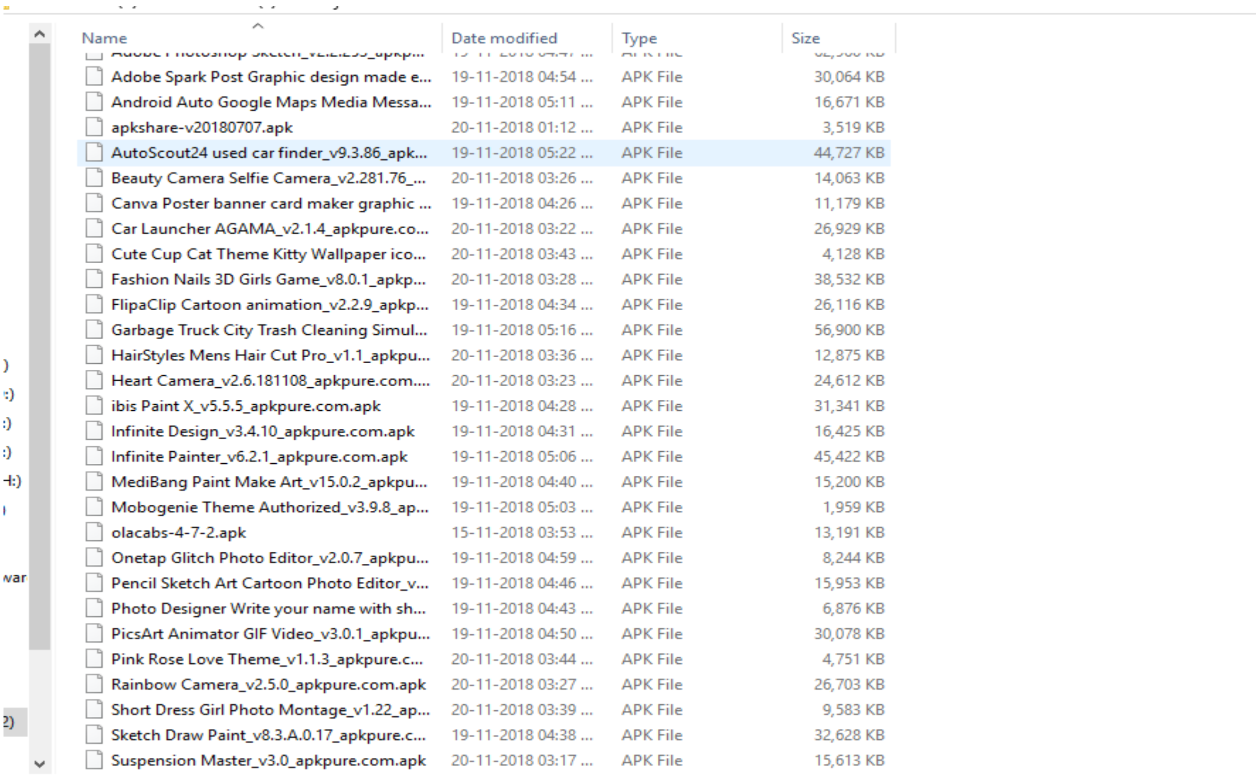
Source: Google Play Store, June 18-Sept. 8, 2014.

Note: The "games" category was expanded to its subcategories. Each of the different subcategories of "games" is marked by an "*." If combined "Games" would make up 11% of total apps. Pew Research Center used the categories available in the Google Play Store and did not do any further categorization. Eight apps did not have category information.

PEW RESEARCH CENTER

Fig 4.1: Categories of permissions as per google developer

2. In the second phase, we studied about the difference between the benign and malicious applications and the permission categories which were listed unsafe to be accessed by each category respectively of an application as was mentioned by Google. Thereby, we worked on a machine learning algorithm model to train our model with both safe and unsafe permission which would help us create a dataset. We downloaded multiple android application samples both malicious and benign to extract the permissions from the manifest.xml file.

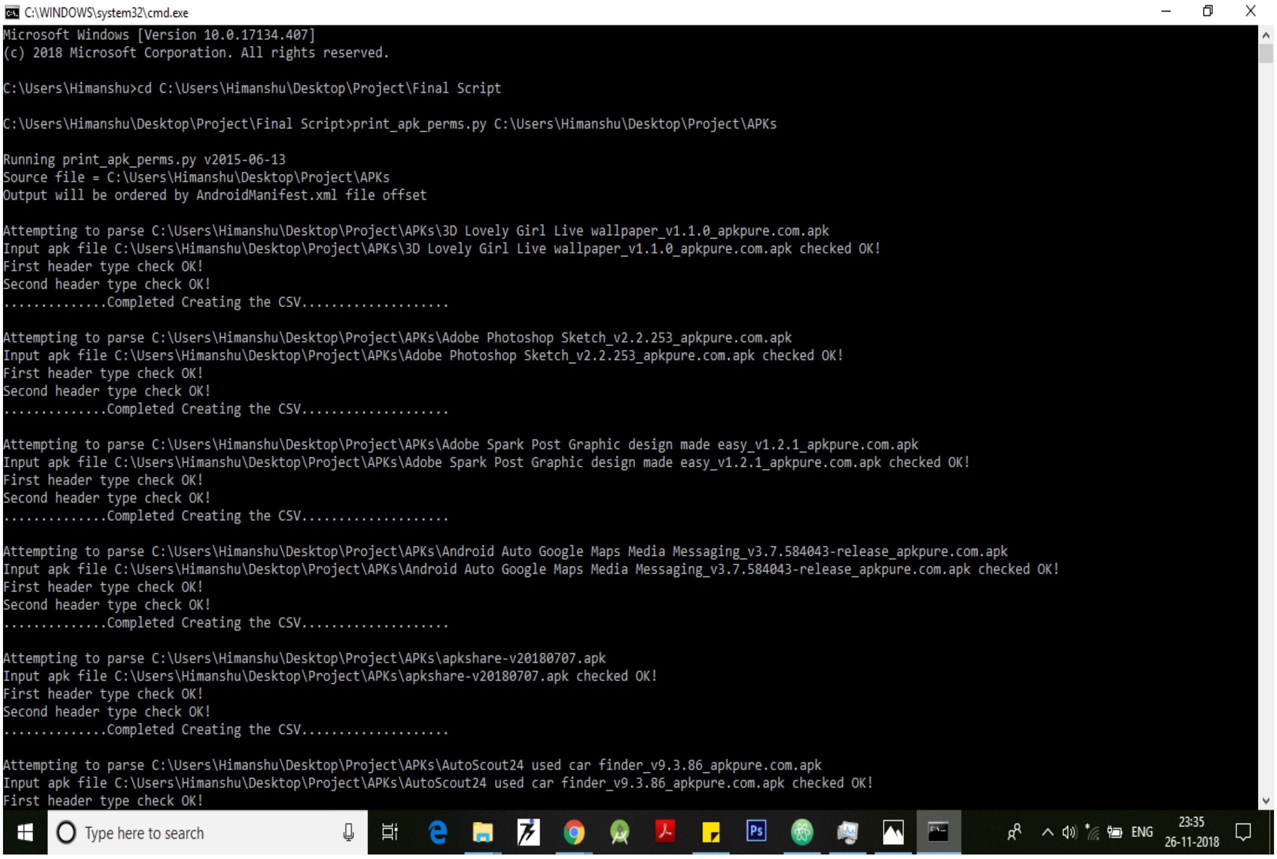


Name	Date modified	Type	Size
Adobe Spark Post Graphic design made e...	19-11-2018 04:54 ...	APK File	30,064 KB
Android Auto Google Maps Media Messa...	19-11-2018 05:11 ...	APK File	16,671 KB
apkshare-v20180707.apk	20-11-2018 01:12 ...	APK File	3,519 KB
AutoScout24 used car finder_v9.3.86_apk...	19-11-2018 05:22 ...	APK File	44,727 KB
Beauty Camera Selfie Camera_v2.281.76_...	20-11-2018 03:26 ...	APK File	14,063 KB
Canva Poster banner card maker graphic ...	19-11-2018 04:26 ...	APK File	11,179 KB
Car Launcher AGAMA_v2.1.4_apkpure.co...	20-11-2018 03:22 ...	APK File	26,929 KB
Cute Cup Cat Theme Kitty Wallpaper ico...	20-11-2018 03:43 ...	APK File	4,128 KB
Fashion Nails 3D Girls Game_v8.0.1_apkp...	20-11-2018 03:28 ...	APK File	38,532 KB
FlipaClip Cartoon animation_v2.2.9_apkp...	19-11-2018 04:34 ...	APK File	26,116 KB
Garbage Truck City Trash Cleaning Simul...	19-11-2018 05:16 ...	APK File	56,900 KB
HairStyles Mens Hair Cut Pro_v1.1_apkpu...	20-11-2018 03:36 ...	APK File	12,875 KB
Heart Camera_v2.6.181108_apkpure.com....	20-11-2018 03:23 ...	APK File	24,612 KB
ibis Paint X_v5.5.5_apkpure.com.apk	19-11-2018 04:28 ...	APK File	31,341 KB
Infinite Design_v3.4.10_apkpure.com.apk	19-11-2018 04:31 ...	APK File	16,425 KB
Infinite Painter_v6.2.1_apkpure.com.apk	19-11-2018 05:06 ...	APK File	45,422 KB
MediBang Paint Make Art_v15.0.2_apkpu...	19-11-2018 04:40 ...	APK File	15,200 KB
Mobogenie Theme Authorized_v3.9.8_ap...	19-11-2018 05:03 ...	APK File	1,959 KB
olacabs-4-7-2.apk	15-11-2018 03:53 ...	APK File	13,191 KB
Onetap Glitch Photo Editor_v2.0.7_apkpu...	19-11-2018 04:59 ...	APK File	8,244 KB
Pencil Sketch Art Cartoon Photo Editor_v...	19-11-2018 04:46 ...	APK File	15,953 KB
Photo Designer Write your name with sh...	19-11-2018 04:43 ...	APK File	6,876 KB
PicsArt Animator GIF Video_v3.0.1_apkpu...	19-11-2018 04:50 ...	APK File	30,078 KB
Pink Rose Love Theme_v1.1.3_apkpure.c...	20-11-2018 03:44 ...	APK File	4,751 KB
Rainbow Camera_v2.5.0_apkpure.com.apk	20-11-2018 03:27 ...	APK File	26,703 KB
Short Dress Girl Photo Montage_v1.22_ap...	20-11-2018 03:39 ...	APK File	9,583 KB
Sketch Draw Paint_v8.3.A.0.17_apkpure.c...	19-11-2018 04:38 ...	APK File	32,628 KB
Suspension Master_v3.0_apkpure.com.apk	20-11-2018 03:17 ...	APK File	15,613 KB

Fig 4.2: The android application samples used for creating dataset.

3. Furthermore, we studied about how the permissions are stored in android package analyser. It leads to the android manifest.xml file which contains all the data regarding the permissions. We created a Java based android package analyser to access the manifest file and extract the permissions usable for the project. But this package analyser worked on a single .apk file at a time. Thus, we created a python script which made it possible to extract permissions from multiple android application samples at the same time and further list them out according to the

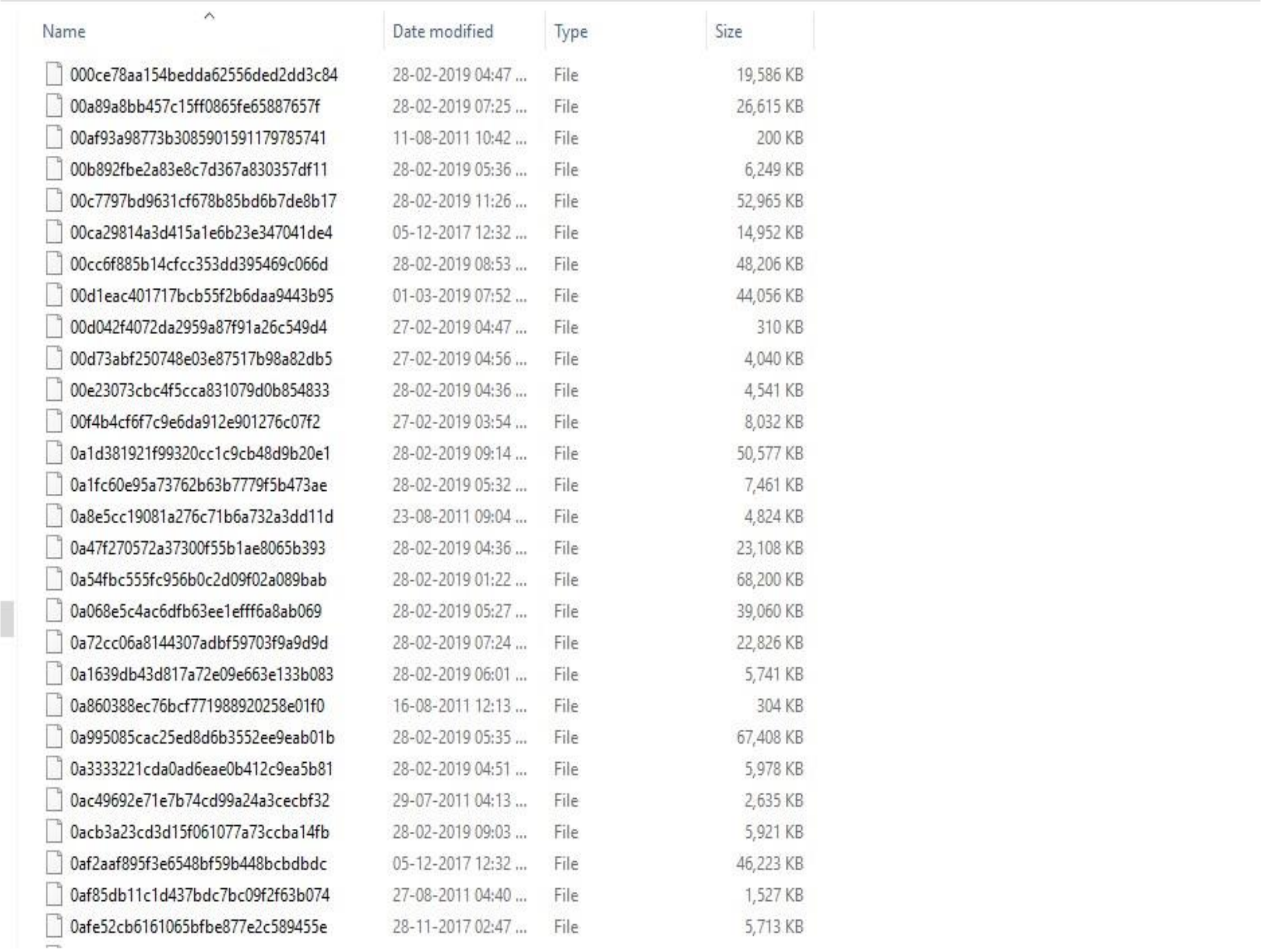
previously created model which was trained with both the safe and unsafe permissions listed by Google.



```
Microsoft Windows [Version 10.0.17134.407]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Himanshu>cd C:\Users\Himanshu\Desktop\Project\Final Script

C:\Users\Himanshu\Desktop\Project\Final Script>print_apk_perms.py C:\Users\Himanshu\Desktop\Project\APKs

Running print_apk_perms.py v2015-06-13
Source file = C:\Users\Himanshu\Desktop\Project\APKs
Output will be ordered by AndroidManifest.xml file offset

Attempting to parse C:\Users\Himanshu\Desktop\Project\APKs\3D Lovely Girl Live wallpaper_v1.1.0_apkpure.com.apk
Input apk file C:\Users\Himanshu\Desktop\Project\APKs\3D Lovely Girl Live wallpaper_v1.1.0_apkpure.com.apk checked OK!
First header type check OK!
Second header type check OK!
.............Completed Creating the CSV...................

Attempting to parse C:\Users\Himanshu\Desktop\Project\APKs\Adobe Photoshop Sketch_v2.2.253_apkpure.com.apk
Input apk file C:\Users\Himanshu\Desktop\Project\APKs\Adobe Photoshop Sketch_v2.2.253_apkpure.com.apk checked OK!
First header type check OK!
Second header type check OK!
.............Completed Creating the CSV...................

Attempting to parse C:\Users\Himanshu\Desktop\Project\APKs\Adobe Spark Post Graphic design made easy_v1.2.1_apkpure.com.apk
Input apk file C:\Users\Himanshu\Desktop\Project\APKs\Adobe Spark Post Graphic design made easy_v1.2.1_apkpure.com.apk checked OK!
First header type check OK!
Second header type check OK!
.............Completed Creating the CSV...................

Attempting to parse C:\Users\Himanshu\Desktop\Project\APKs\Android Auto Google Maps Media Messaging_v3.7.584043-release_apkpure.com.apk
Input apk file C:\Users\Himanshu\Desktop\Project\APKs\Android Auto Google Maps Media Messaging_v3.7.584043-release_apkpure.com.apk checked OK!
First header type check OK!
Second header type check OK!
.............Completed Creating the CSV...................

Attempting to parse C:\Users\Himanshu\Desktop\Project\APKs\apkshare-v20180707.apk
Input apk file C:\Users\Himanshu\Desktop\Project\APKs\apkshare-v20180707.apk checked OK!
First header type check OK!
Second header type check OK!
.............Completed Creating the CSV...................

Attempting to parse C:\Users\Himanshu\Desktop\Project\APKs\AutoScout24 used car finder_v9.3.86_apkpure.com.apk
Input apk file C:\Users\Himanshu\Desktop\Project\APKs\AutoScout24 used car finder_v9.3.86_apkpure.com.apk checked OK!
First header type check OK!
```

Fig 4.3: Creating CSV file from dataset using python script

4. After the successful implementation if various machine learning algorithm on the created dataset, we went deeper into the more complex algorithms using deep learning. Firstly, we focused on the implementation of the Convolutional neural network which takes input as images and extracts features and parameters to give out the output and in our case, to classify the Android Applications into benign or malicious. For this the Android Application are to be renamed into unique hash file name to avoid duplicacy. We created a python script to uniquely identify the MD5 and rename the Android Application into its respective MD5 name. If this python script encounters any duplicacy, it gives the name of the Android Application along with its respective MD5 and deletes the duplicate files. Thus ensuring unique files.



Name	Date modified	Type	Size
000ce78aa154bedda62556ded2dd3c84	28-02-2019 04:47 ...	File	19,586 KB
00a89a8bb457c15ff0865fe65887657f	28-02-2019 07:25 ...	File	26,615 KB
00af93a98773b3085901591179785741	11-08-2011 10:42 ...	File	200 KB
00b892fbe2a83e8c7d367a830357df11	28-02-2019 05:36 ...	File	6,249 KB
00c7797bd9631cf678b85bd6b7de8b17	28-02-2019 11:26 ...	File	52,965 KB
00ca29814a3d415a1e6b23e347041de4	05-12-2017 12:32 ...	File	14,952 KB
00cc6f885b14cfcc353dd395469c066d	28-02-2019 08:53 ...	File	48,206 KB
00d1eac401717bcb55f2b6daa9443b95	01-03-2019 07:52 ...	File	44,056 KB
00d042f4072da2959a87f91a26c549d4	27-02-2019 04:47 ...	File	310 KB
00d73abf250748e03e87517b98a82db5	27-02-2019 04:56 ...	File	4,040 KB
00e23073cbc4f5cca831079d0b854833	28-02-2019 04:36 ...	File	4,541 KB
00f4b4cf6f7c9e6da912e901276c07f2	27-02-2019 03:54 ...	File	8,032 KB
0a1d381921f99320cc1c9cb48d9b20e1	28-02-2019 09:14 ...	File	50,577 KB
0a1fc60e95a73762b63b7779f5b473ae	28-02-2019 05:32 ...	File	7,461 KB
0a8e5cc19081a276c71b6a732a3dd11d	23-08-2011 09:04 ...	File	4,824 KB
0a47f270572a37300f55b1ae8065b393	28-02-2019 04:36 ...	File	23,108 KB
0a54fbc555fc956b0c2d09f02a089bab	28-02-2019 01:22 ...	File	68,200 KB
0a068e5c4ac6dfb63ee1efff6a8ab069	28-02-2019 05:27 ...	File	39,060 KB
0a72cc06a8144307adbf59703f9a9d9d	28-02-2019 07:24 ...	File	22,826 KB
0a1639db43d817a72e09e663e133b083	28-02-2019 06:01 ...	File	5,741 KB
0a860388ec76bcf771988920258e01f0	16-08-2011 12:13 ...	File	304 KB
0a995085cac25ed8d6b3552ee9eab01b	28-02-2019 05:35 ...	File	67,408 KB
0a3333221cda0ad6eae0b412c9ea5b81	28-02-2019 04:51 ...	File	5,978 KB
0ac49692e71e7b74cd99a24a3cecbf32	29-07-2011 04:13 ...	File	2,635 KB
0acb3a23cd3d15f061077a73ccba14fb	28-02-2019 09:03 ...	File	5,921 KB
0af2aaf895f3e6548bf59b448bcbdbdc	05-12-2017 12:32 ...	File	46,223 KB
0af85db11c1d437bdc7bc09f2f63b074	27-08-2011 04:40 ...	File	1,527 KB
0afe52cb6161065bfbe877e2c589455e	28-11-2017 02:47 ...	File	5,713 KB

Fig 4.4: Renaming of the Android Application into MD5 hash name

5. In addition to this, the unique hash file name also helps in extracting other properties of the Android Applications. Furthermore, these Android Applications which are renamed into their respective MD5 are to be converted into the images. For this, we created a python script using multiple python libraries such as PIL, imageio etc, to convert the Android Application to images. The Android Applications are converted into strings and the strings are further converted to binary strings of 0 or 1. Thus, using these binary strings, the strings are organized into 2D array. Then the respective array can be seen as a greyscale image which is in the range of [0,255].

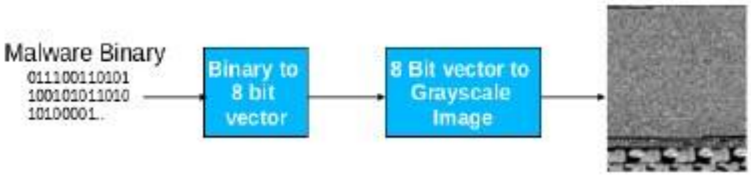


Fig 4.5: Flow diagram to show conversion of Android Application to greyscale images.

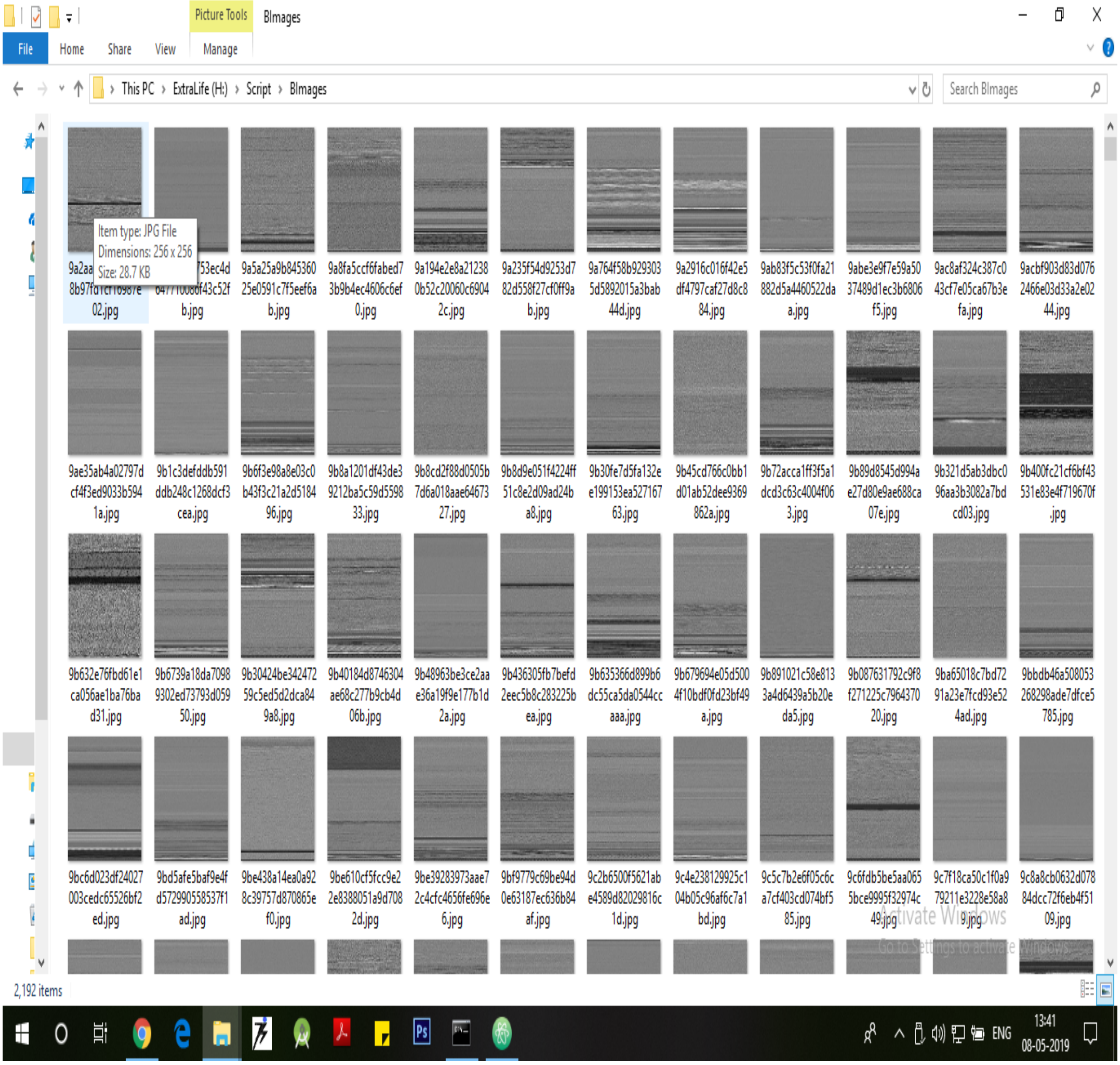

Picture Tools
BImages
File
Home
Share
View
Manage
This PC › ExtraLife (H:) › Script › BImages
Search BImages
Item type: JPG File
Dimensions: 256 x 256
Size: 28.7 KB
9a2aa 8b97fd1cf16987e02.jpg
753ec4d 647710086f43c52fb.jpg
9a5a25a9b84536025e0591c7f5eef6ab.jpg
9a8fa5ccf6fabed73b9b4ec4606c6ef0.jpg
9a194e2e8a212380b52c20060c69042c.jpg
9a235f54d9253d782d558f27cf0ff9ab.jpg
9a764f58b9293035d5892015a3bab44d.jpg
9a2916c016f42e5df4797caf27d8c884.jpg
9ab83f5c53f0fa21882d5a4460522daa.jpg
9abe3e9f7e59a5037489d1ec3b6806f5.jpg
9ac8af324c387c043cf7e05ca67b3efa.jpg
9acbf903d83d0762466e03d33a2e0244.jpg
9ae35ab4a02797dcf4f3ed9033b5941a.jpg
9b1c3defddb591ddb248c1268dcf3cea.jpg
9b6f3e98a8e03c0b43f3c21a2d518496.jpg
9b8a1201df43de39212ba5c59d559833.jpg
9b8cd2f88d0505b7d6a018aae6467327.jpg
9b8d9e051f4224ff51c8e2d09ad24ba8.jpg
9b30fe7d5fa132ee199153ea52716763.jpg
9b45cd766c0bb1d01ab52dee9369862a.jpg
9b72acca1ff3f5a1dcd3c63c4004f063.jpg
9b89d8545d994ae27d80e9ae688ca07e.jpg
9b321d5ab3dbc096aa3b3082a7bdcd03.jpg
9b400fc21cf6bf43531e83e4f719670f.jpg
9b632e76fbd61e1ca056ae1ba76bad31.jpg
9b6739a18da70989302ed73793d05950.jpg
9b30424be34247259c5ed5d2dca849a8.jpg
9b40184d8746304ae68c277b9cb4d06b.jpg
9b48963be3ce2aae36a19f9e177b1d2a.jpg
9b436305fb7befd2eec5b8c283225bea.jpg
9b635366d899b6dc55ca5da0544ccaaa.jpg
9b679694e05d5004f10bdf0fd23bf49a.jpg
9b891021c58e8133a4d6439a5b20eda5.jpg
9b087631792c9f8f271225c796437020.jpg
9ba65018c7bd7291a23e7fcd93e524ad.jpg
9bbdb46a508053268298ade7dfce5785.jpg
9bc6d023df24027003cedc65526bf2ed.jpg
9bd5afe5baf9e4fd572990558537f1ad.jpg
9be438a14ea0a928c39757d870865ef0.jpg
9be610cf5fcc9e22e8388051a9d7082d.jpg
9be39283973aae72c4cfc4656fe696e6.jpg
9bf9779c69be94d0e63187ec636b84af.jpg
9c2b6500f5621abe4589d82029816c1d.jpg
9c4e238129925c104b05c96af6c7a1bd.jpg
9c5c7b2e6f05c6ca7cf403cd074bf585.jpg
9c6fdb5be5aa0655bce9995f32974c49.jpg
9c7f18ca50c1f0a979211e3228e58a89.jpg
9c8a8cb0632d07884dcc72f6eb4f5109.jpg
Activate Windows
Go to Settings to activate Windows.
2,192 items
ENG
13:41
08-05-2019

Fig 4.6: The dataset of greyscale images.

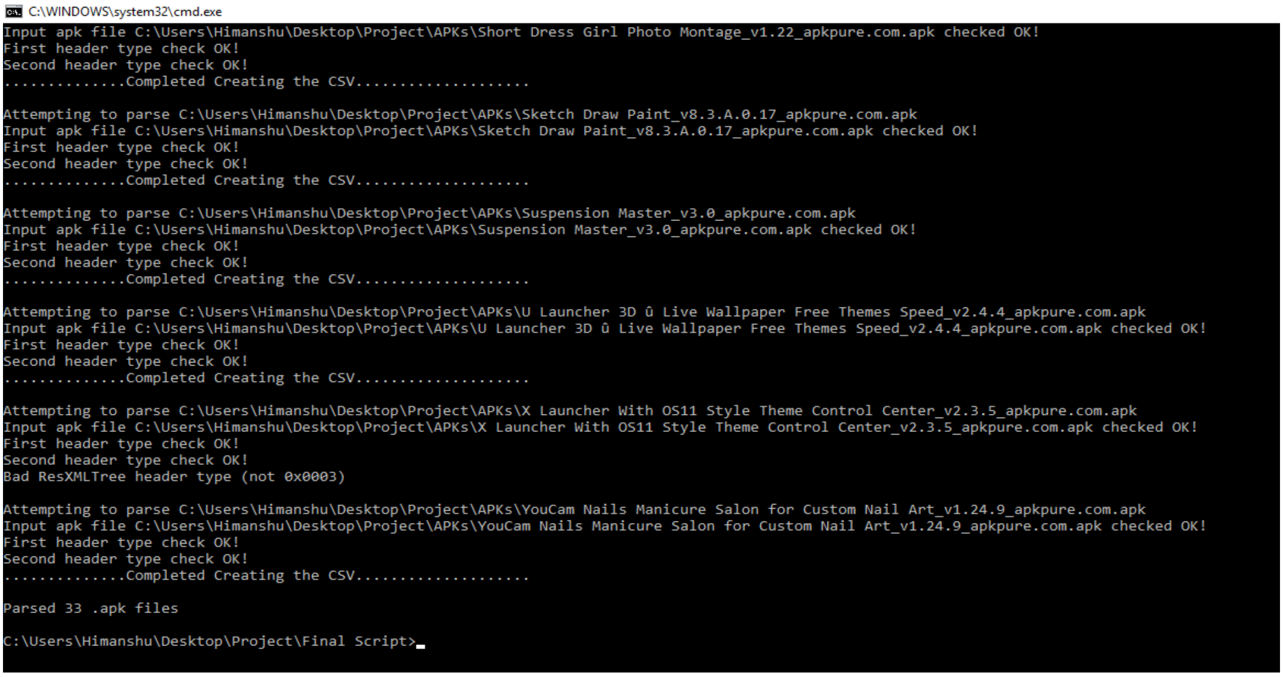
6. After the successful conversion of Android Applications into images, we observe that the images obtained were of different sizes and dimensions. In order to get more accurate results and better understanding of the images, the images are to be normalised to be of same dimension and used as the input. The dimensions of the greyscale image used in the project is 256 * 256.

Chapter-5

TEST PLAN

In this chapter, we focused on working out our algorithms and scripts on the project to create a dataset and the machine learning algorithms and to carry out comparative analysis and store the results for further analysis.

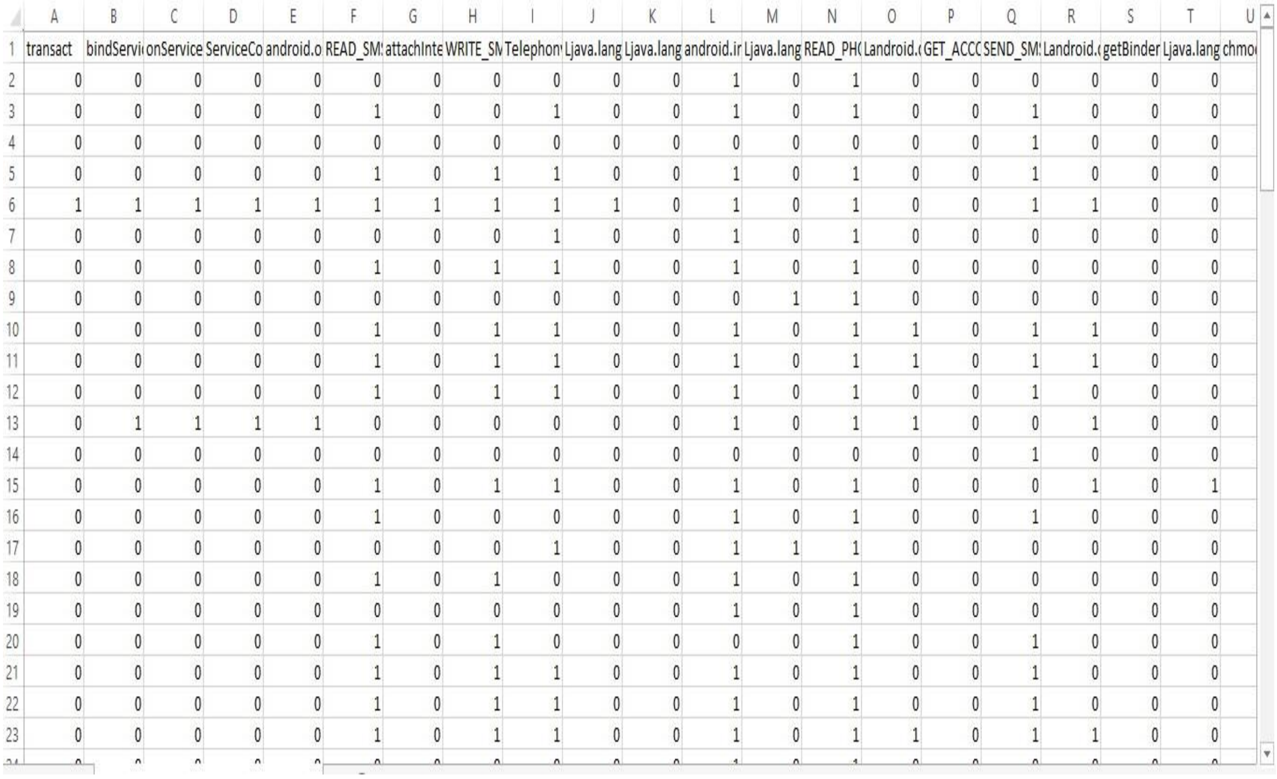
At first we run the script which extracts the above mentioned permissions and list them into a CSV file format making it easier to run machine learning algorithms on the dataset created.



```
Input apk file C:\Users\Himanshu\Desktop\Project\APKs\Short Dress Girl Photo Montage_v1.22_apkpure.com.apk checked OK!
First header type check OK!
Second header type check OK!
..............Completed Creating the CSV...................

Attempting to parse C:\Users\Himanshu\Desktop\Project\APKs\Sketch Draw Paint_v8.3.A.0.17_apkpure.com.apk
Input apk file C:\Users\Himanshu\Desktop\Project\APKs\Sketch Draw Paint_v8.3.A.0.17_apkpure.com.apk checked OK!
First header type check OK!
Second header type check OK!
..............Completed Creating the CSV...................

Attempting to parse C:\Users\Himanshu\Desktop\Project\APKs\Suspension Master_v3.0_apkpure.com.apk
Input apk file C:\Users\Himanshu\Desktop\Project\APKs\Suspension Master_v3.0_apkpure.com.apk checked OK!
First header type check OK!
Second header type check OK!
..............Completed Creating the CSV...................

Attempting to parse C:\Users\Himanshu\Desktop\Project\APKs\U Launcher 3D û Live Wallpaper Free Themes Speed_v2.4.4_apkpure.com.apk
Input apk file C:\Users\Himanshu\Desktop\Project\APKs\U Launcher 3D û Live Wallpaper Free Themes Speed_v2.4.4_apkpure.com.apk checked OK!
First header type check OK!
Second header type check OK!
..............Completed Creating the CSV...................

Attempting to parse C:\Users\Himanshu\Desktop\Project\APKs\X Launcher With OS11 Style Theme Control Center_v2.3.5_apkpure.com.apk
Input apk file C:\Users\Himanshu\Desktop\Project\APKs\X Launcher With OS11 Style Theme Control Center_v2.3.5_apkpure.com.apk checked OK!
First header type check OK!
Second header type check OK!
Bad ResXMLTree header type (not 0x0003)

Attempting to parse C:\Users\Himanshu\Desktop\Project\APKs\YouCam Nails Manicure Salon for Custom Nail Art_v1.24.9_apkpure.com.apk
Input apk file C:\Users\Himanshu\Desktop\Project\APKs\YouCam Nails Manicure Salon for Custom Nail Art_v1.24.9_apkpure.com.apk checked OK!
First header type check OK!
Second header type check OK!
..............Completed Creating the CSV...................

Parsed 33 .apk files

C:\Users\Himanshu\Desktop\Project\Final Script>_
```

Fig 5.1: Python script to create a dataset

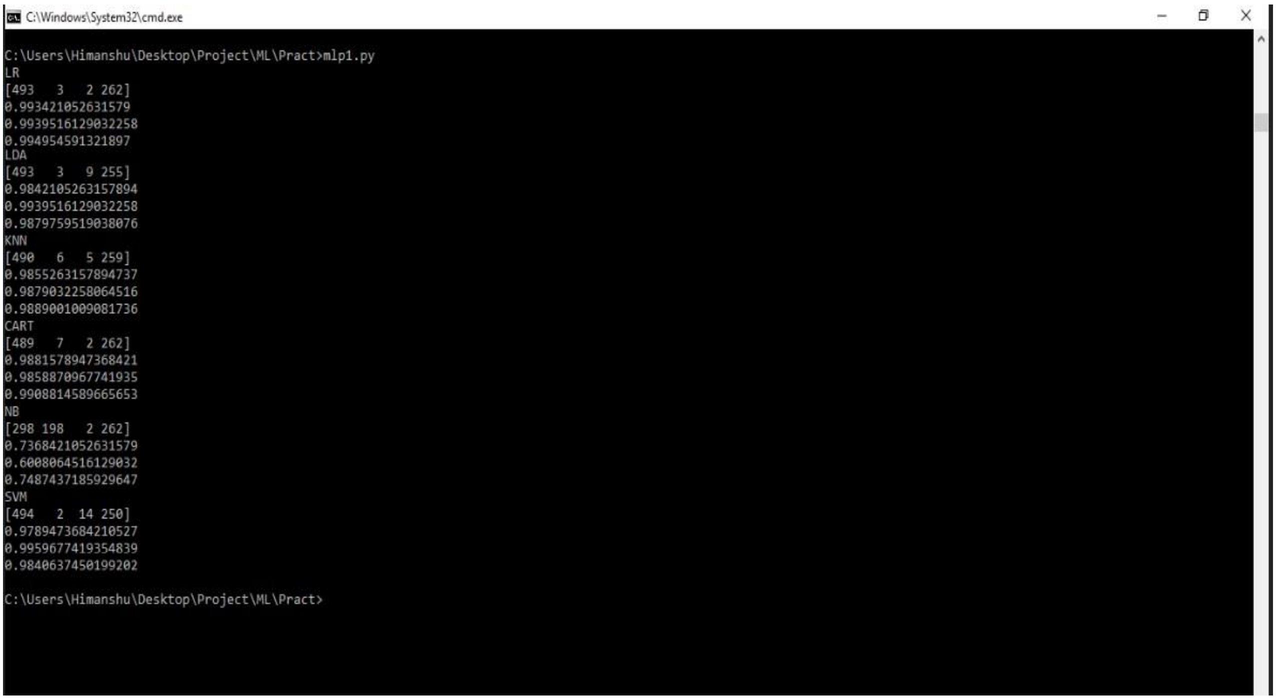
With the help of this script, we parsed more than 4000 android application samples both benign and malicious and extracted permissions which was stored in a CSV file format and used as a dataset. The dataset was further divided in the ratio of 20:80. The 80% of the dataset was used to train the model and to identify the benign and malicious permissions and to create Labelled analysis. The remaining 20% of the dataset was utilized for the testing phase of the algorithm and to give out the accuracy measures and other parameters to finalize the algorithm for the project.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	transact	bindServi	onService	ServiceCo	android.o	READ_SM:	attachInte	WRITE_SN	Telephon	Ljava.lang	Ljava.lang	android.ir	Ljava.lang	READ_PH(	Landroid.(	GET_ACCC	SEND_SM:	Landroid.(	getBinder	Ljava.lang	chmo(
2	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	
3	0	0	0	0	0	1	0	0	1	0	0	1	0	1	0	0	1	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
5	0	0	0	0	0	1	0	1	1	0	0	1	0	1	0	0	1	0	0	0	
6	1	1	1	1	1	1	1	1	1	1	0	1	0	1	0	0	1	1	0	0	
7	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	
8	0	0	0	0	0	1	0	1	1	0	0	1	0	1	0	0	0	0	0	0	
9	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	
10	0	0	0	0	0	1	0	1	1	0	0	1	0	1	1	0	1	1	0	0	
11	0	0	0	0	0	1	0	1	1	0	0	1	0	1	1	0	1	1	0	0	
12	0	0	0	0	0	1	0	1	1	0	0	1	0	1	0	0	1	0	0	0	
13	0	1	1	1	1	0	0	0	0	0	0	1	0	1	1	0	0	1	0	0	
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
15	0	0	0	0	0	1	0	1	1	0	0	1	0	1	0	0	0	1	0	1	
16	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0	1	0	0	0	
17	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	
18	0	0	0	0	0	1	0	1	0	0	0	1	0	1	0	0	0	0	0	0	
19	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	
20	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	0	1	0	0	0	
21	0	0	0	0	0	1	0	1	1	0	0	1	0	1	0	0	1	0	0	0	
22	0	0	0	0	0	1	0	1	1	0	0	1	0	1	0	0	1	0	0	0	
23	0	0	0	0	0	1	0	1	1	0	0	1	0	1	1	0	1	1	0	0	

Fig 5.2: Created dataset

Therefore, we studied about various machine learning algorithms specially the supervised learning algorithms as we were working on the labelled dataset and got the following observations as result.



```
C:\Users\Himanshu\Desktop\Project\ML\Pract>mlp1.py
LR
[493   3   2 262]
0.993421052631579
0.9939516129032258
0.994954591321897
LDA
[493   3   9 255]
0.9842105263157894
0.9939516129032258
0.9879759519038076
KNN
[490   6   5 259]
0.9855263157894737
0.9879032258064516
0.9889001009081736
CART
[489   7   2 262]
0.9881578947368421
0.9858870967741935
0.9908814589665653
NB
[298 198   2 262]
0.7368421052631579
0.6008064516129032
0.7487437185929647
SVM
[494   2  14 250]
0.9789473684210527
0.9959677419354839
0.9840637450199202

C:\Users\Himanshu\Desktop\Project\ML\Pract>
```

Fig 5.3: Results obtained after running various ML algorithms.

Chapter-6

RESULTS AND ANALYSIS

After the Implementation of the python script to extract the required permissions and to create the dataset and application of various supervised machine learning algorithms to the dataset created, we obtain certain results that gives out the measure of accuracy of the algorithms listed below-

1. Linear Regression
2. K- Nearest Neighbors
3. Decision Trees
4. Naïve Bayes
5. Support Vector Machines

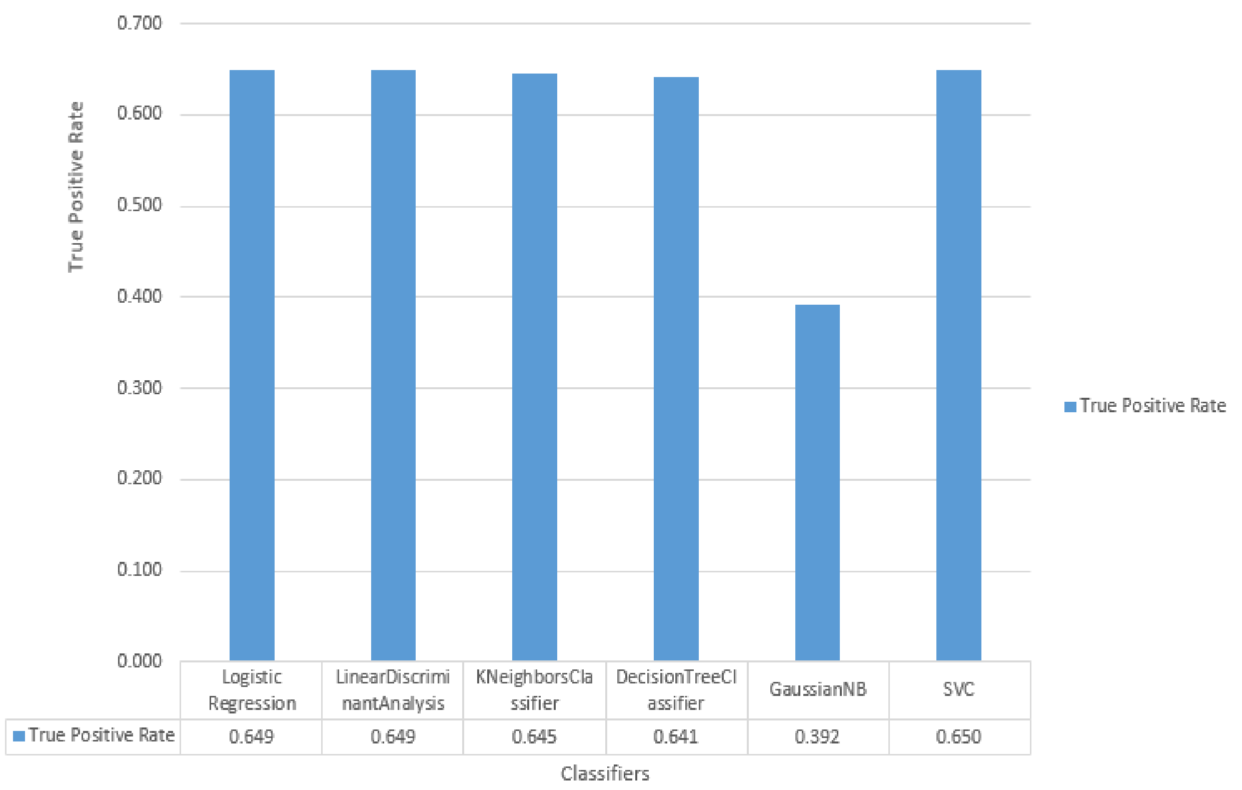
The above mentioned algorithms were tested on various parameters and features such as True positives, True Negatives, False Positives and False Negatives were calculated and the measures such as Accuracy, Precision were calculated.

Classifier	TPR	FPR	FNR	TNR	Accuracy	Precision	F1 Score
Logistic Regression	0.649	0.004	0.003	0.345	99.34%	99.40%	99.50%
Linear Discriminant Analysis	0.649	0.004	0.012	0.336	98.42%	99.40%	98.80%
K Neighbors Classifier	0.645	0.008	0.007	0.341	98.55%	98.79%	98.89%
Decision Tree Classifier	0.641	0.012	0.005	0.342	98.82%	98.59%	99.09%
Gaussian NB	0.392	0.261	0.003	0.345	73.68%	60.08%	74.87%
SVC	0.650	0.003	0.018	0.329	97.89%	99.60%	98.41%

Table 6.1: Results of comparative analysis

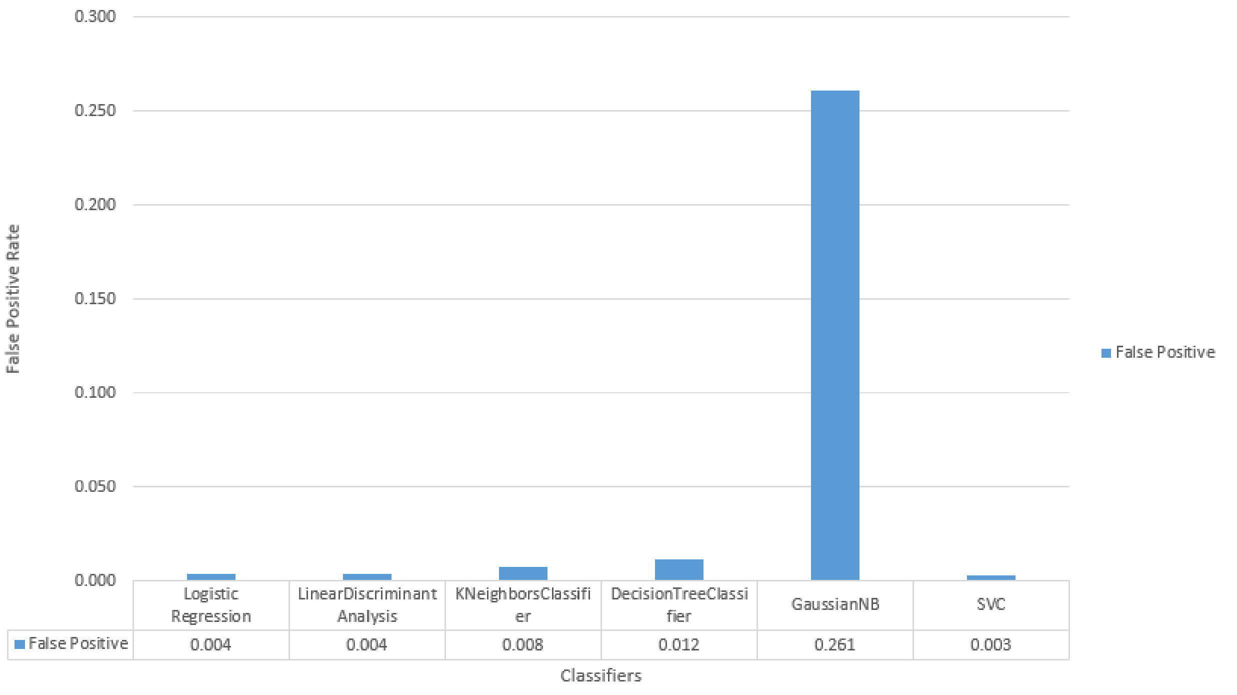
According to the results obtained, we concluded that out of all the algorithms that were tested with our dataset, the algorithm **LOGISTIC REGRESSION** gives out the most accurate results with an accuracy measure of **99.34%** and the precision and f1 score of **99.40% & 99.50%** respectively.

According to the results obtained, we do graphical comparative analysis of finding the best suitable method for our project. Following are the graphical representation of the various parameters-



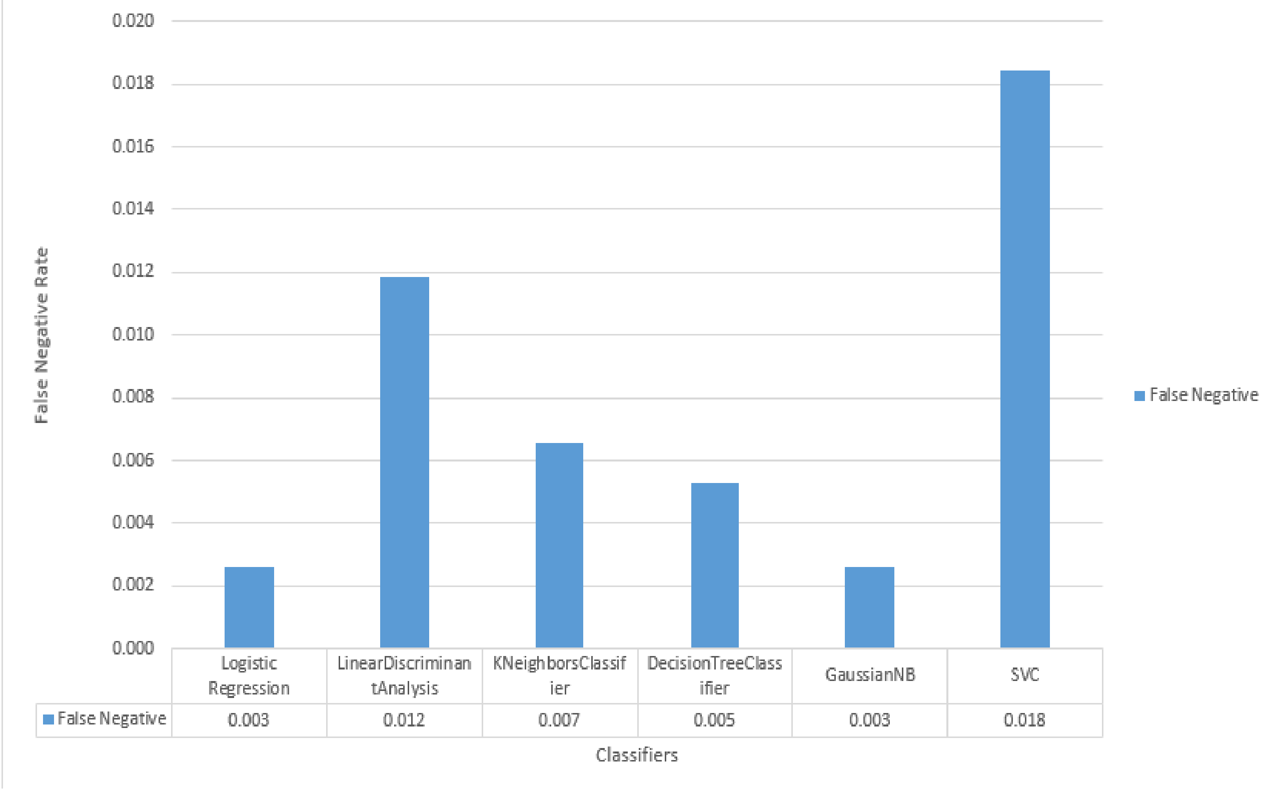
Graph 6.1: Graphical representation of TPR of various ML algos.

The above graph shows the comparative analysis of the first feature of comparison that is True positive which is true if the output of both the actual class and predicted class is true.



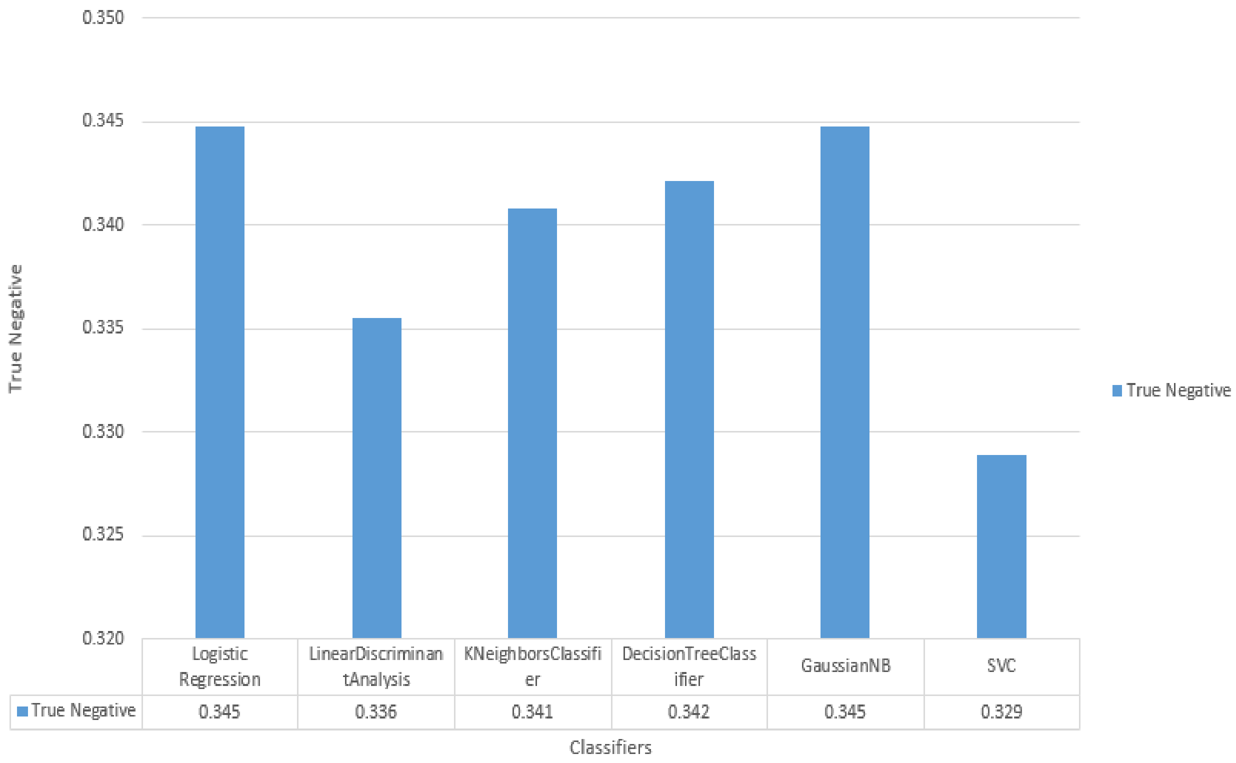
Graph 6.2: Graphical representation of FPR of various ML algos.

The above graph shows the comparative analysis of second feature of comparison that is False Positive which is true when the output value of the actual class is false and predicted class is true.



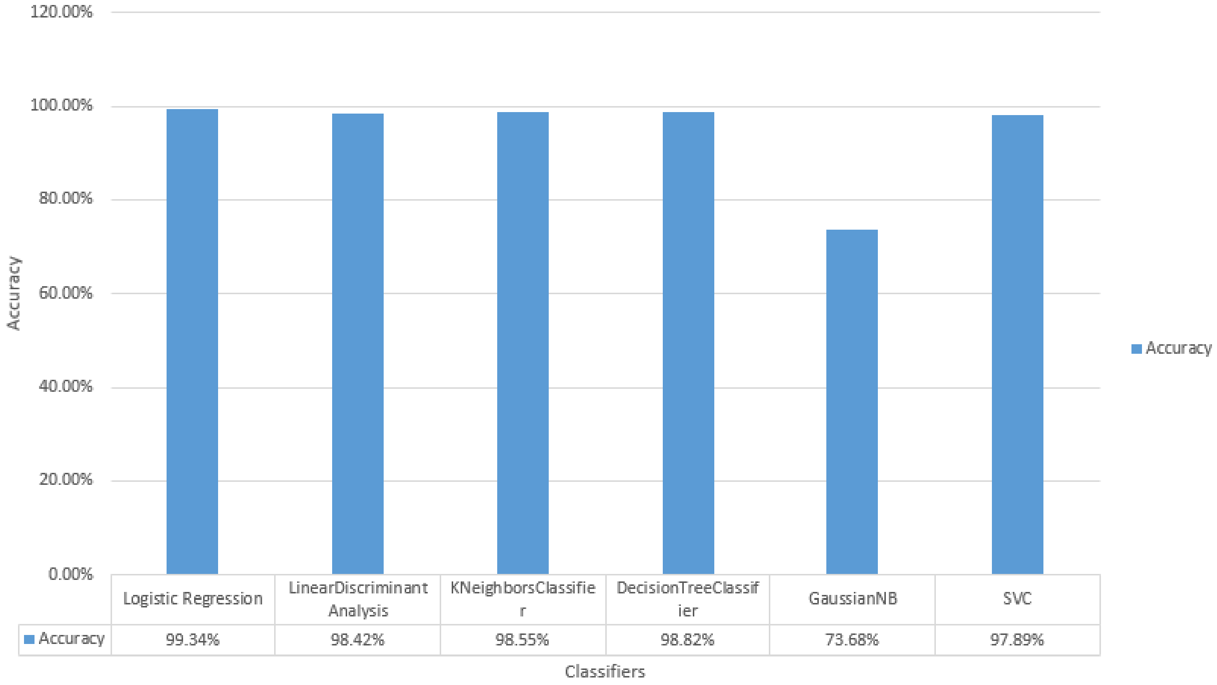
Graph 6.3: Graphical representation of FNR of various ML algos.

The above graph shows the comparative analysis of third feature of comparison that is False Negative which is true is the output value of the actual class is false and the predicted class is false.



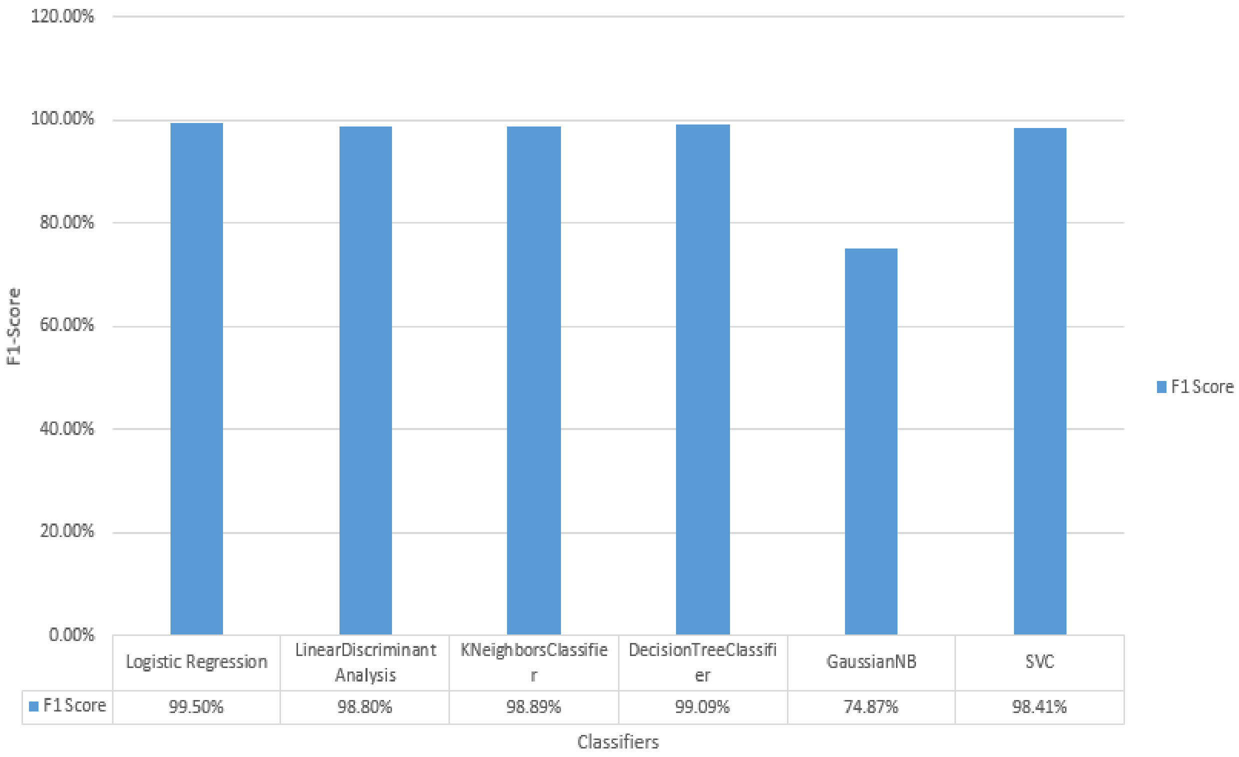
Graph 6.4: Graphical representation of TNR of various ML algos.

The above graph shows the comparative analysis of fourth feature of comparison that is True Negative which is true if the output value of both the actual class and the predicted class is false.



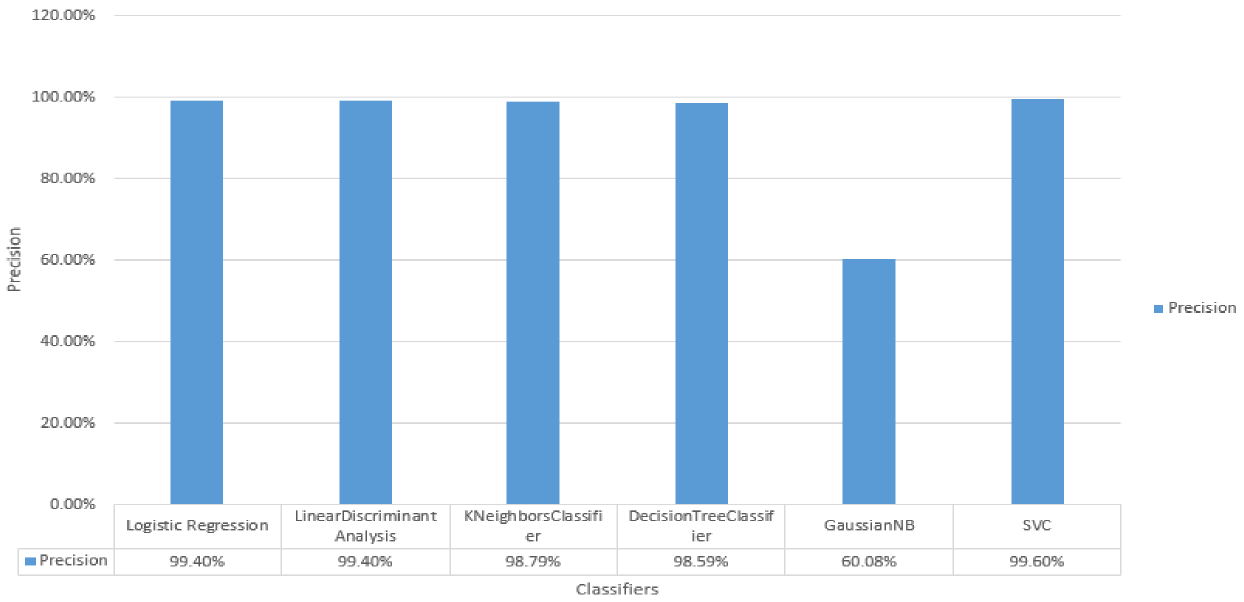
Graph 6.5: Graphical representation of Accuracy of various ML algos.

The above graph shoes the comparative analysis of the first parameter that is accuracy which is calculate by the ratio of count of correctly predicted observation to the count of total observations.



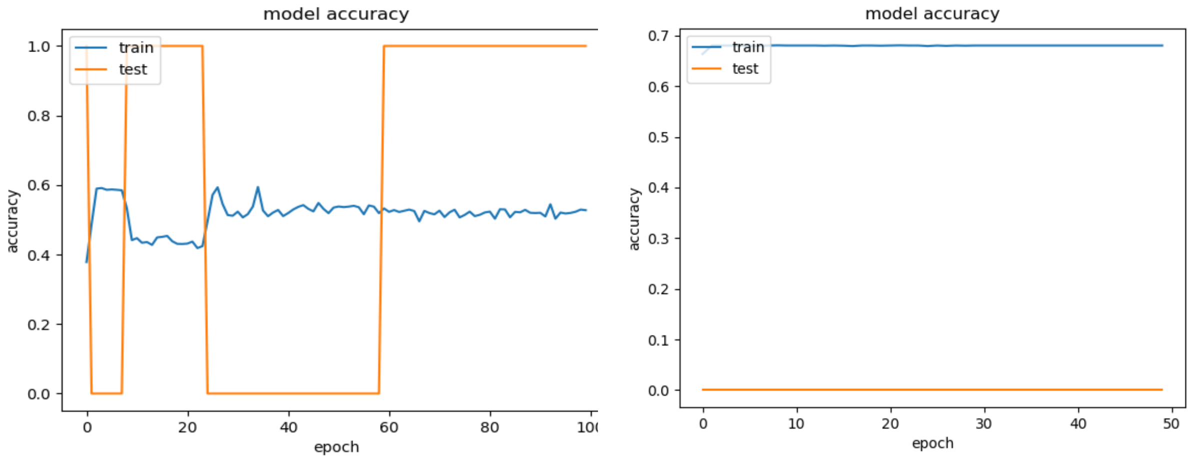
Graph 6.6: Graphical representation of F1 score of various ML algos.

The above graph shows the comparative analysis of the second parameter that is F1 score which is calculated by the weighted average of the value of precision and the value of recall.



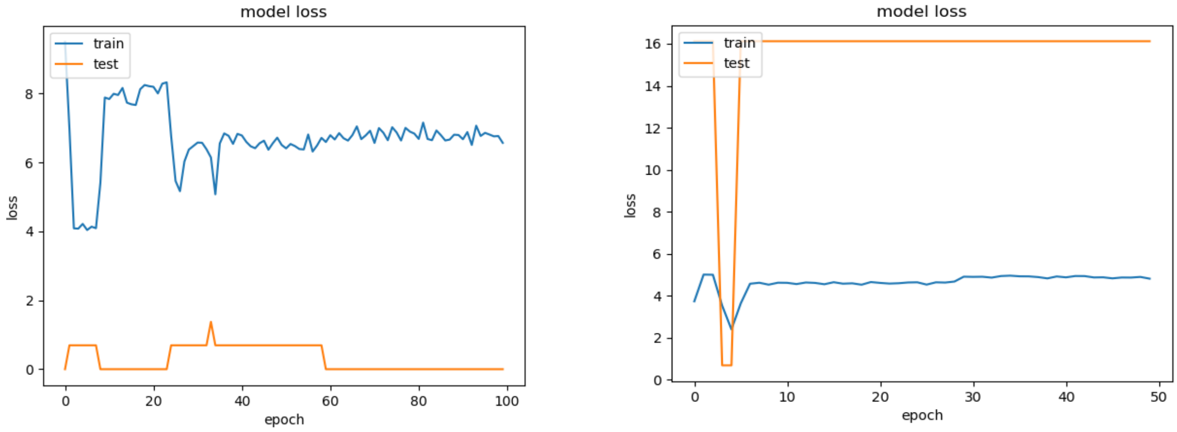
Graph 6.7: Graphical representation of Precision of various ML algos.

The above graph shows the comparative analysis of the third parameter that is precision which is calculated by the ratio of the output value of the predicted output which are correct to the total predicted output which are positive.



Graph 6.7: Graphical representation of accuracy by convoluting 2 × 2 filter matrix and 3 × 3 filter matrix

The above graph shows the comparison of accuracy of two filters, that is, 2 × 2 & 3× 3. The first figure shows the average accuracy of 59% for 2 × 2 filter matrix of the training data and the second image shoes the accuracy of 69% for 3× 3 filter matrix of training data. So we can conclude that if the size of the filter is increased than the accuracy of the model increases. On the other hand, if the filter size is increased, the training log loss decreases that is, for the first filter (2 × 2) log loss is 0.63 and for (3× 3) the log loss is 0.42.



Graph 6.7: Graphical representation of loss by convoluting 2× 2 filter matrix and 3× 3 filter matrix.

Chapter-7

CONCLUSION

In our project, we have implemented various machine learning algorithms especially the supervised learning for detection of malware or anomaly in the android application samples and classified them into two groups namely benign and malicious. For this classification, we use a labelled dataset which was created using permissions extracted from multiple android applications from manifest file which was accessed programmatically using a python script which was made by reverse engineering the android package analyser. We were able to create a labelled dataset and thus used supervised learning algorithms on the dataset by splitting the dataset into training and testing set. With these dataset, we calculated the values of different parameters and thus compared the parameters and features for various supervised machine learning algorithms and concluded that the algorithm which gives the largest accuracy values is the Logistic Regression. Thus with an accuracy value of **99.34%** and the precision value of **99.40%**, it is the most suitable algorithm for our current dataset. After obtaining the above results, it is best to say that we obtained no normal samples which gave misclassified results whereas the malicious samples showed very minimal results. After the exploration of the convolutional neural network in classification of Android application into benign or malicious by converting them into images and using different filters on the input. We concluded that the convolutional neural network gives the accuracy percentage of 59% for the filter of 2× 2 blocks of pixels and 69% for the filter of 3× 3 blocks of pixels. In conclusion, machine learning algorithms especially logistic regression works better for the dataset.

FUTURE SCOPE

After studying the concepts of various machine learning algorithms and after application of such machine learning algorithms on the dataset created by extracting the permissions from the android manifest file and comparing the results of these algorithms on the basis of various parameters such as accuracy, precision etc. we saw great potential in the machine learning algorithms in detection of malware and helping the android community to have safer experience. The dataset created by the python script is a self-created dataset and can be used in future research and used to implement other complex algorithms to get better outcomes or same algorithms can be implemented with a different approach to improve the time and space complexity of the models. Furthermore, more features and parameters can be included to enhance the analysis of performance measure. In our project, we only considered the permissions for the classification of android application samples, to extend the research we can include other comparison methods like intend call, system calls etc.

REFERENCES

[1] D. Stevens, 2007. http://blog.didierstevens.com/2007/10/23/a000n0000-0000o000l00d00-0i000e000-00t0r0000i0000c000k/.

[2] Schmidt Aubery-Derrick, (2011). Detection of Smart Phone Malware. Unpublished PhD. Thesis Electronic and Information Technology University Berlin. pp. 1-211.

[3] Christodorescu Mihai & Jha Somesh, (2003). Static Analysis of Executables to Detect Malicious Patterns. In Proceedings of the 12th conference on USENIX Security Symposium - Volume 12, SSYM'03, pp. 12, Berkeley, CA, USA, 2003.

[4] Raymond W. Lo, Karl N. Levitt & Ronald A. Olsson. (1995). MCF: A Malicious Code Filter. Computers and Security, 14(6), pp. 541 – 566.

[5] Bryan Dixon, Yifei Jiang, Abhishek Jaiantilal, & Shivakant Mishra, (2011). Location based Power Analysis to Detect Malicious Code in Smartphones. In Proceedings of the 1st ACM workshop on Security and Privacy in Smartphones and Mobile Devices, SPSM '11, pp. 27–32.

[6] Hahnsang Kim, Joshua Smith & Kang G. Shin, (2008). Detecting Energy Greedy Anomalies and Mobile Malware Variants. In Proceedings of the 6th international conference on Mobile Systems, Applications, and Services, MobiSys '08, pp. 239–252.

[7] Lei Liu, Guanhua Yan, Xinwen Zhang & Songqing Chen, (2009). Virusmeter: Preventing your Cell Phone from Spies. In Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection, RAID '09, pp. 244–264.

[8] Tchakounté F. & Dayang P. (2013). System Calls Analysis of Malwares on Android. International Journal of Science and Technology 2(9), pp. 669-674.

[9] Burquera I., Zurutuza U. & Nadjm-Tehrani S., (2011). Crowdroid: Behavior-based Malware Detection System for Android. In Proceedings of the 1st ACM workshop on Security and Privacy in Smartphones and Mobile Devices, pp. 15-26.

[10] Liang Xie, Xinwen Zhang, Jean-Pierre Seifert, & Sencun Zhu, (2010). PBMDS: A Behavior-based Malware Detection System for Cell Phone Devices. In Proceedings of the third ACM conference on Wireless network security, WiSec '10, pp. 37–48. International Journal of Network Security & Its Applications (IJNSA) Vol.7, No.6, November 2015 34

[11] Abhijit Bose, Xin Hu, Kang G. Shin, & Taejoon Park, (2008). Behavioral Detection of Malware on Mobile Handsets. In Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services, MobiSys '08, pp. 225–238.

[12] Yerima S.Y., Sezer S. & McWilliams G., (2014). Analysis of Bayesian Classification Based Approaches for Android Malware Detection. IET Information Security, Volume 8, Issue 1, January 2014, p. 25 – 36, DOI: 10.1049/iet ifs.2013.0095.

[13] Zico J. Kolter & Marcus A., (2006). Maloof: Learning to Detect and Classify Malicious Executables in the Wild. J. Mach. Learn. Res., 7, pp. 2721–2744, December, 2006.

[14] Matthew G. Schultz, Eleazar Eskin, Erez Zadok & Salvatore J. Stolfo, (2001). Data Mining Methods for Detection of New Malicious Executables. In Proceedings of the 2001 IEEE Symposium on Security and Privacy, SP'01, pp. 38–, Washington, DC, USA, 2001. IEEE Computer Society.

[15] Tesauro G.J., Kephart J.O., & Sorkin G.B., (1996). Neural Networks for Computer Virus Recognition. IEEE Expert, 11(4), pp.5-6.

[16] Asaf S., Uri K., Yuval E., Chanan G. & Yael W., (2011). Andromaly: A Behavioural Malware Detection Framework for Android Devices. Journal of Intelligent Information Systems, pp. 1-30. doi: 10.1007/s10844-010-0148-x.

[17] Schmidt Aubery-Derrick, Jan Hendrik Clausen, Seyit Ahmet Camtepe & Sahin Albayrak, (2009). Detectiong Symbian OS Malware through Static Function Calls Analysis. In Proceedings of the 4th IEEE International Conference on Malicious and unwanted Software (Malware 2009), pp. 1522, IEEE, 2009.

[18] Schmidt Aubery-Derrick, Frank Peters, Florian Lamour & Sahin Albayrak, (2008). Monitoring Smartphones for Anomaly Detection. In Proceedings of the 1st International Conference on MOBILe Wireless MiddleWARE, Operating Systems and Applications (MOBILEWARE '08), pp.16, ICST, Brussels, Belgium, Belgium, 2008.

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: 10/May/19

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | [✓ B.Tech Project Report] | Paper

Name: HIMANSHU KUSHWAHA, ADITYA KAPOOR Department: CSE Enrolment No 151361, 151359

Contact No. 8629010355 E-mail. aditya.kapoor33@gmail.com

Name of the Supervisor: DR. EKTA GANDOTRA

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): MALICIOUS ANDROID APPLICATION DETECTION USING MACHINE LEARNING

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages = 61
- Total No. of Preliminary pages = 10
- Total No. of pages accommodate bibliography/references = 2

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at 15 (%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

Ekta 10/5/19.

(Signature of Guide/Supervisor)

Signature of HOD

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
10.05.2019	• All Preliminary Pages • Bibliography/Images/Quotes • ~~14~~ 20 Words String	5%	Word Counts	9,014
Report Generated on			Character Counts	46,858
11.05.2019		Submission ID	Total Pages Scanned	50
		1128686243	File Size	4.77M

SBhatt

Checked by
Name & Signature Ashok

Chandan 11.05.2019
Librarian

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com