INDEX

TITLES Page. No

Abstract 1

1. Introduction 2
2. Literature Survey 7
3. System Analysis
   1. Existing System 13
   2. Proposed System 13
   3. Feasibility Study
      1. Economical Study 14
      2. Technical Study 14
      3. Social Study 15
   4. System Requirements 16
   5. System Architecture 17
   6. Data Flow Diagram 21
   7. UML Diagram 23
   8. Input Design 25
   9. Output Design 26
4. Implementation
   1. Modules 27
   2. Software Environment
      1. Python 28
      2. Why Choose Python 29
   3. Sample Code
      1. SVC Algorithm 65
      2. Main Program 66
5. System Testing & Output Screens
   1. System Testing 74
      1. Unit Testing 76
      2. Integration Testing 76
      3. Acceptance Testing 77
      4. Test Cases 77
   2. Output Screens 78
6. Conclusion & Future Work
   1. Conclusion 81
   2. Future Work 82
7. Biblography 83

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No** | **Figure Name** | **Page. No** |
| 1 | Android Stack Structure | 18 |
| 2 | Data Flow Diagram | 22 |
| 3 | Use Case Diagram | 23 |
| 4 | Use Case Diagram | 24 |
| 5 | DBN Architecture | 41 |
| 6 | Flow Chart | 43 |
| 7 | Machine Learning With Python | 45 |
| 8 | Machine Leaning Model | 48 |
| 9 | Machine Learning Problem | 52 |
| 10 | Python in Artificial Intelligence | 57 |
| 11 | Support Vector Machine | 65 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| Table No | Table Name | Page. No |
| 1 | Types of Android Malware | 19 |
| 2 | Difference Between Machine Learning and Deep Learning | 42 |
| 3 | Test Cases | 77 |

**Android Malware Detection Using Machine Learning Classifiers**

**ABSTRACT**

**Abstract**

Android malware growth has been increasing dramatically along with increasing the diversity and complicity of their developing techniques. Machine learning techniques are the current methods to model patterns of static features and dynamic behaviors of Android malware. Whereas the accuracy rates of the machine learning classifiers increase with increasing the quality of the features, we relate between the apps’ features and the features that are needed to deliver its category’s functionality. Differently, our classification approach defines legitimate static features for benign apps under a specific category as opposite to identifying malicious patterns. We utilize the features of the top rated apps in a specific category to train a malware detection classifier for that given category. Android apps stores organize apps into different categories, for instance, 26 categories on Google Play Store. Each category has its distinct functionalities which means the apps under a specific category are similar in their static and dynamic features. In general, benign apps under a certain category tend to share a common set of features. On the contrary, malicious apps tend to request abnormal features, less or more than what is common for the category that they belong to. This study proposes category- based machine learning classifiers to enhance the performance of classification models at detecting malicious apps under a certain category. The intensive machine learning experiments proved that category-based classifiers report a remarkable higher average performance compared to non-category based.

1

**INTRODUCTION**

**Introduction**

According to International Data Corporation (IDC), Android OS is the most popular smartphone platform with 82.2% of the market share of smartphones, while 13.9% for iOS apple in the second quarter of 2015 [[3](#_bookmark104)]. Statistically speaking, it is also the first targeted platform by malware authors seeking to take the control over millions of Android smartphones over the world. Due to the popularity of Android’s smartphones, its apps’ security is a serious issue concerning 80% of smartphones users.

Android is an open source development environment that offers a rich SDK that enables developers to deploy their own apps and distribute them through Android apps centers. Android’s popularity is a result of being an open source, third-party distribution centers, a rich SDK, and the popularity of Java as a programming language. Importantly, due to this open environment, malware authors can develop malicious apps that abuse the features that the platform offers or pack a legitimate app with a piece of malicious code; besides, exploiting vulnerabilities in the platform, hardware, or other installed apps to lunch malicious behaviors. Mainly, malware authors seek access confidential data of a device’s user, monetary benefits via premium SMS, or joining the device to a botnet. Even legitimate apps introduce the risk of privacy-invading; Mcafee reported in Q1 2014 that 82% of Android apps track user’s and 80% gather location data.

Research studies in the Android malware detection field work in three approaches static, dynamic or hybrid. In static analysis, malware is disassembled into a source code from where specific features are extracted. In dynamic analysis, malware is monitored at run-time in a virtual environment. In the both approaches, machine learning algorithms have been used to build classification models by training classifiers with datasets

of malware features that collected from static or dynamic analysis. The learned classification models are then used to detect malicious apps and classify them into their families.

In this study, we approach the problem differently by utilizing the features of benign apps for malware detection. We relate between the features that the app requests and the common features for its category.

2

Android apps stores organize apps into different categories; for example, Google play store organizes apps in 26 categories such as: ”Health & Fitness”, ”News & Magazine”, ”Books & References”, ”Music & Audio”, etc. Each category has its distinct functionalities which means the apps under a certain category share similar features. One group of these features are the permissions; per- missions are the privileges that enable apps to access the system’s resources to perform their functions. Each built-in permission is responsible for providing the capabilities to execute a particular process. Apps belong to a specific category deliver the same functionality as a result they require a common combination of permissions. For instance, apps under ”Communication” category commonly request READ CONTACTS but it is un- common if it is requested by apps under ”News & Magazines”. In general, benign apps under a certain category tend to have a common set of features: permissions, intents filters, hardware components, broadcast receivers, APIs, etc. On the contrary, malicious apps tend to request abnormal features, less or more than what is common for the category that they belong to. Repeatedly from that point of view, this study proposes category-based machine learning classifiers to enhance the performance of classification models at detecting malicious apps under a certain category. .

## Motivations

Android malware growth has been increasing dramatically along with increasing of the diversity and complicity of their developing techniques. According to F-Secure, a computer security company, Android had the biggest share of smartphone malware by 97% in 2014 [[9](#_bookmark108)]. Android global market share of smartphone industry is 78% which rep- resents the biggest share among other smartphone platforms [[3](#_bookmark104)]. Statically speaking, Android apps’ security concerns one billion of active users over the world [[8](#_bookmark107)]. Due to the openness of Android environment, there is a remarkable increase in the number of published Android apps. According to the statistics (staista), the number of available apps to download in Google play stores was around 1,500,000 in 2014 [[2](#_bookmark103)].

Static analysis can report a high accuracy rate in detection malware; and it is relatively cheap compared to dynamic analysis in terms of effort, time, and computational resources. Machine learning algorithms are the current methods for detecting malware on Android. In the both approaches, static and dynamic, they are used to model pat- terns of static features and dynamic behaviors of malware.

3

Mostly, the researches focus on training supervised machine learning classifiers to classify the malware to a known malware family, or using semi-supervised learning to discover a new one.

In fact, machine learning techniques can report remarkable accuracy rates at detecting malicious apps depending on the quality of the features that used for training the classifiers e.g. how specific they are. Whereas the accuracy rates of the classifiers increase with increasing the quality of the features, we relate between the apps’ features and the features that are needed to deliver its category’s functionality to detect malicious patterns. In other words, we train a malware detection classifier for each category, separately.

## Scope

This study is a static analysis that uses the features that can be extracted from the source codes of the apps’ .apk files. We parsed three group of features from each apps in our datasets: permissions, broadcast receivers, and APIs. Whereas we pro- pose category-based classifiers to improve the performance of the classification models at detecting malicious apps under a certain category, we worked on two categories on the Google Play Store ”Music & Audio” and ”Personalization”. We built three datasets of apps features: apps from all categories (allCateg), apps from ”Music & Audio” category (musicCateg), and apps from ”Personalization” category (personaCateg); in each dataset the benign apps were downloaded from the top rated apps on the Google Play Store while malicious apps from virushare. For each dataset, we trained three ma- chine learning classifiers: Support Vector Machines, Random Forest, and AdaBootsM. We tested the classifiers with two datasets of apps from ”Music & Audio” and ”Personalization” categories. By evaluating the performance of the classifiers, the category-based classifiers reported a higher performance by 3.5-4.5% compared to non-category based.

4

## Contribution

Mostly, research studies in Android malware detection focus on identifying the features of malicious apps by using machine learning techniques to recognize and model the malicious patterns of static features and dynamic behaviors of malware. Up to our knowledge, no researches have worked on relating between the apps’ features and the features of benign to distinguish benign form malicious ones in the same category. Differently, our classification approach defines legitimate static features for benign apps under a specific category as opposite to identifying malicious patterns. We utilize the

features of the top rated apps in a specific category to define a profile of the common sets of features for that category. In other words, to detect whether or not the app posses the characteristics of benign, we relate between the app’s features and the features that are needed to deliver its category’s functionalities. Android stores organize apps into different categories; 26 categories on the Google Play Store, for example. In each category, the apps deliver a similar functionality as a result the they tend to request a common set of features like same permissions, APIs, hardware components, broadcast receivers, intents filters,..etc. On the contrary, malicious apps tend to have abnormal features, less or more than what is common for the category that they belong to. Whereas the accuracy rates of the classifiers increase with increasing the quality of the features, we propose category-based classifiers to enhance the performance of machine learning algorithms at detecting malicious apps under a given category.

**Background**

Android is an open-source operating system for mobile phones, tablets, TVs, cars, embedded and wearable devices. It was built based on Linux kernel, developed by Google and released on September 23, 2008 . As a result of the open environment of Android, many companies and manufacturers uses it as a platform for their products. Besides, this environment allows companies to customize the Android system to fit with their devices needs. Android offers a friendly development environment through a variety of tools: Android SDK, Android NDK, Android Debug Bridge (ADB), and Android Developer Tools (Eclipse). Android Software Development Kit (SDK) is updated with every release of a new version of Android; it provides comprehensive packages of Java framework classes, libraries, and debuggers for programmers. Also, the SDK offers An- droid Emulator which enables developers to run and test their apps on different virtual devices that run different versions of Android OS.

5

Secondly, Android Native Development Kit (NDK) is a set of libraries written in C, C++, and other languages can be loaded into Java code through System.loadLibrary call. Android Debug Bridge (ADB) is a command line tool in a client-server form that consists of three components: client, server, and daemon. The client runs on the development machine where the daemon runs in the background on each emulator or device instance and the server manages communication between the client and the daemon. ADB enables developers to test their apps for bugs by connecting the device running the software to a PC and using terminal commands. Finally, Eclipse is the official Integrated Development Environment (IDE) to develop Android apps. It provides many features through GUI or command lines; it allows programmers to develop their apps with different programming languages: JAVA is widely used for Android apps, C, C++. Google Play Store is the official distribution center for Android Apps which are developed by Google or third-parties. It allows Android users to browse, install, and update the apps. Over 50 billion apps were downloaded by Android users from Google Play Store in 2013 . Unsurprisingly, the open development environment of Android encourages the developers and even the attackers to deploy their own applications. Android markets follow specific procedures that aim to detect and remove malicious apps. For instance, Google Play Store uses a tool called Bouncer that scans the uploaded apps and applies security measurements before publishing. The Google Bouncer is a dynamic analysis technique that tests apps by running them in virtual environments to monitor automatically the app’s behavior. Even with the security measurements that Google takes, the attackers find their ways to pass their malicious apps through the scanning system by using hiding techniques such as encryption and heuristic evasion. Security researchers from Columbia University have exploited vulnerabilities in Google’s Bouncer system. The vulnerabilities allow the attackers to pass the malware apps to the Android market [[6](#_bookmark106)]. The team found that the examined dynamic and static analysis tools were vulnerable to repackaged and heuristic evasion based malware. Indeed, Android malware detection is a significant issue for security researchers; also, it is a serious challenge for Android users’ privacy. The following sections briefly cover basics of Android OS which include its architecture, security features, apps components, permissions model and permission protection levels.

6

**LITERATURE SURVEY**

**Literature Review**

The initial studies on smart phone malware were chiefly targeted on understanding the threats behaviors of rising malware. There has been vital work on the matter of police work malware on mobile devices. Many approaches monitor the facility usage of applications and report abnormal consumption. Others monitor system calls and arrange to discover uncommon system call patterns. Different approaches additional ancient comparison with acknowledged malware or different heuristics. Signatures primarily based ways, introduced within the mid-90s area unit ordinarily employed in malware detection. The main weakness of this kind of approach is its weakness in police work metamorphic and unseen malware. Rather than victimization predefined signatures for malware detection, data processing and machine learning techniques give a good thanks to dynamically extract malware patterns. For smart phone-based mobile computing platforms, recent years have witnessed an increasing range of additional sophisticated malware attacks like repackaging. Recent analysis consistently characterizes existing mechanical man malware from varied aspects, together with their installation ways, activation mechanism moreover because the nature of carried malicious payloads. supported the analysis with four representative mobile security software package over 1200 collected malware, their experiments show the weakness of current malware detection solutions and need the necessity to develop next-generation antimobile-malware solutions. One existing work has used data processing and options generated from windows workable API calls. They achieved sensible leads to a really giant scale dataset with concerning 35,000 transportable workable files. Another activity foot printing methodology additionally provides a dynamic approach to discover self-propagating malware. All these existing ways have basically advanced the mechanical man malware detection; however the misuse detection isn't reconciling to the novel mechanical man malware and continually needs frequent change of the signatures. Here lies the analysis gap.

In comparison, our work is motivated by a number of the higher than techniques and approaches, however with a spotlight on developing straightforward and effective malware detection approaches, while not looking forward to advanced dynamic runtime analysis and any static predefined malware signatures.

7

Our objective is to mix permissions and API calls as options to characterize malware and use

machine learning techniques to mechanically extract patterns to differentiate benign and malicious

Apps. This study may be a static analysis that uses the options that may be extracted from the

supply codes of the app’s .apk files

After conducting a thorough literature review of the research in the area of Android malware detection, I observed different types of objectives of such research. Many research articles are focused on surveying existing methods of solving the malware detection problem. These articles do a systematic review of different techniques that other researchers have used for this purpose and compare the results.

Liu et al. (2020) review in detail different approaches and research status from different perspectives like sample acquisition, data preprocessing, feature selection, machine learning algorithms, and performance evaluation. Finally, Odusami et al. (2018) review existing malware detection methods, including static and dynamic analysis approaches, describe the strengths and weaknesses of each approach, and conclude that machine learning-based methods show the best detection accuracy and thus are promising for the future.

Some studies are focused on choosing the correct feature set. The features used are just as crucial to the end outcome of the malware detection exercise as the techniques and algorithms used to perform the detection. So, these studies provide valuable insight into the right feature set to use. Wenetal. (2017) use a combination of features from both static and dynamic analysis, then apply PCA to reduce the dimensionality of data and use SVM to perform the classification of applications into benign and malware classes. Roy et al. (2020) build a feature extraction module that performs static analysis to map each API call to certain features. Then, feature vectors are generated, and dimensionality is reduced, following which classification algorithms are used to differentiate between benign and malicious applications.

Daoudietal. (2021) convert the byte code of the application into grey-scale vector images and use 1- dimensional Convolutional Neural Networks to detect malware. This approach circumvents the need for creating comprehensive hand-crafted features and uses the raw byte code of the application for analysis. Jiang et al. (2020) study the permissions frequently used by malicious applications and identify permissions they call dangerous fine-grained permissions, which better differentiate benign and malicious applications. These features are then used in machine learning models to perform the classification.

8

Other studies focus more on optimizing the detection algorithm than the feature set. These studies are focused on improving the detection performance by choosing the right machine learning algorithm and/or using various techniques to enhance the performance of traditional algorithms. Rathore et al., 2021 use multiple types of static analysis features and compare the performance of different machine learning and deep learning techniques, both supervised and unsupervised. They found that the baseline Random Forest model without any feature reduction achieved the best performance. Shao et al., 2021 extract features from the Android application package, use the relief feature selection method to select features,

use different sampling strategies to address the class imbalance, and improve traditional ensemble bagging algorithms to achieve the best performance.

There are also studies in the literature that use a combination of multiple techniques instead of applying one technique to improve results. These studies tend to develop a complex approach but can possibly lead to better results. Yerima et al. (2018) use a combination of machine learning algorithms for increased the performance of the detection system. They first perform classification using base classifiers and then re-classify the base classifier predictions using ranking-based algorithms to achieve the final prediction. Almin et al. (2015) analyze permissions requested by applications at the time of installation and perform a combination of clustering and classification to detect malware.

In a study conducted by Syrris and Geneiatakis (2021), the authors start by appreciating how much the Android operating system has, over the years, been advancing in enhancing its robustness. The robustness is associated with the advanced technologies, significant community support, and availability of tons of resources on the internet. However, all these privileges come at a cost on source platforms in which security is compromised. In this regard, malicious applications find a way to bypass some security protocols. In addressing this problem, Syrris and Geneiatakis (2021) state that there are several approaches that can be used to leverage machine learning to detect malware through the help of static analysis data. According to Kumar et al., 2018 statistical analysis and feature extraction are the two main methodologies that support machine learning in malware detection processes.

In research that was conducted by Li et al. (2018), the authors indicated that new malicious Android applications are introduced into the mobile ecosystem every ten seconds.

9

This statistic is worrying, and there are chances of interfering with the mobile ecosystem growth globally if something is not done. Further, the authors acknowledge that there is a need for this problem to be addressed before it affects the integrity of the Android software engineering processes. In combating the problem, the authors acknowledge the need to have a scalable malware detection approach based on the dynamics of the mobile ecosystem and the development of Android applications.

The advancements in terms of technology in developing android operating systems have created more opportunities like the existence of e-commerce, among others. However, it has led to more challenges like cyber-attacks. Among the challenges posed by android devices and the mobile ecosystem as a whole, malicious applications have undoubtedly taken the lead (Christiana et al., 2020). The malware has also continued to advance in terms of sophistication and intelligence such that it has become hard for them to be easily detected through the existing systems. For instance, signature-based systems used for malware detection have become inefficient in detecting advanced malware applications (Christiana et al., 2020). As a result, machine learning techniques are now at the top in dealing with this challenge.

Cybercrimes are rapidly increasing on Android-based devices because of their wide usage across the globe. This increase has made it possible for malicious individuals to engineer malicious applications for their gains and at the expense of the users. In this research, the authors concur with Christiana et al. (2020) that the deployment of machine learning techniques is the option needed for curbing malicious android applications. Malicious applications can be classified using machine learning models to differentiate between benign and malicious android applications (Sharma et al., 2020). Additionally, a comparative analysis aimed at calculating the computational time necessary to detect malicious applications is a requirement necessary in machine learning techniques.

The challenge posed by this mechanism is based on the fact that some large bundle applications cannot be easily scaled hence creating the need for Significant Permission IDentification (SigPID). This approach has an efficiency of about 93.62 percent of malware detection in a particular dataset, making it the best method to detect malware. SigPID uses permission usage to analyze the increasing number of humanoid malware. It is not necessary for the engineers to analyze all humanoid permissions for them to detect the existence of malware (Assisi et al., nd).

10

Instead, mining the permission information is critical in determining the most important permissions that can easily lead to the classification of malicious and benign applications, hence complementing machine learning in malware detection. Kyaw and Kham (2019) reiterate that a scalable malware detection method that yields optimum results is the use of permission analysis through the SigPID approach. Malware applications are thus identifiable through the analysis of the permission behavior. Additionally, pruning procedures are therefore necessary for identifying the most significant permissions that will provide the desired results through the multilevel pruning methodology.

Android applications are readily available because of the comprehensive community support and other open sources that make it easier for malware engineers to develop more malicious applications. Android devices have been the target for malware applications because of the worldwide reception of the android devices (Kumar et al., 2018). For the purpose of reaching high levels of accuracy when detecting malware, a small subset of specific features should be considered. Furthermore, Android is actively implementing new security controls, including the use of a unique user ID (UID) and system permissions for every application (Ranaetal., 2018). Therefore, the use of machine learning classifiers has become one of the best approaches for detecting any android malware in the mobile ecosystem and other android devices. Fallah and Bidgoly (2019) research demonstrated the need to benchmark machine learning algorithms before the associated techniques can achieve the required level of efficacy. The basis of this approach is identifying the family of particular malicious applications. Moreover, the authors demonstrate the need to use combined techniques to get optimum results that are consistent across the platforms based on the selected datasets or classifications of the malware. In this context, the authors recommend using machine learning and network-based detection techniques. In the case of machine learning, the detection process should work in both the unsupervised and supervised machine learning methods to get viable results that will be used in the decision-making process. However, this research does not clearly demonstrate how machine learning algorithms will handle the new variants of malware that have not been tested through the existing algorithms. This means that for the machine learning techniques to be effective, the algorithms have to be updated every often to capture and detect new families of malware.

11

The rising attacks on smart phones result from Android being the most used OS. In this context, the authors give a reason why it has been so easy for attackers to use malicious Android applications to launch attacks. The prompts posed to the user when installing the applications that require them to accept all sorts of necessary permissions before the installation are the main issue. Before a user gives the needed permissions, some applications fail to install, leaving the user with no option but to provide the permissions (Singh et al., 2022). Consequently, some Android applications are not approved by the associated organizations and might be misused in collecting user data that might eventually be misused. For this reason, the application of machine learning algorithms has increasingly been used in detecting Android malware. Android classification algorithms like decision trees, vector machines, and random forests form the basis of machine learning success in detecting malware on Android devices.

Considering the above types of literature available, I try to combine two types of studies in this project. The project's goal is to determine which feature set works best and explore different detection algorithms to determine which works best. It should be noted that the combination of multiple detection techniques is not in the scope of this project.

12

**SYSTEM ANALYSIS & DESIGN**

**3.1 Existing system**

One existing work has used data processing and options generated from windows workable

API calls. They achieved sensible leads to a really giant scale dataset with concerning 35,000

transportable workable files. Another activity foot printing methodology additionally provides a

dynamic approach to discover self-propagating malware. All these existing ways have basically

advanced the mechanical man malware detection; however, the misuse detection isn't reconciling

to the novel mechanical man malware and continually needs frequent change of the signatures.

Here lies the analysis gap. In exiting system they implemented the classifiers like naive bayes and

decision tree which gives the poor accuracy

**Disadvantages**

misuse detection isn't reconciling

accuracy is less

mechanical man malware detection

**3.2 proposed system**

in the proposed system we implement a better feature extraction techniques and then we apply the

genetic algorithm for feature extraction and then we use two machine learning model called as SVM

and multi perception classifier for classification of android malware detection which gives the better accuracy ratio when compare to existing system

**Advantages**

easy to identify and block malware

accuracy is more

dynamic feature extraction using genetic algorithm

13

**3.3 FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

**3.3.1 ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### 3.3.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

14

**3.3.3 SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with

it. His level of confidence must be raised so that he is also able to make some constructive criticism,

which is welcomed, as he is the final user of the system.

15

**3.4 SYSTEM REQUIREMENTS:**

**HARDWARE REQUIREMENTS:**

* System : Pentium IV 2.4 GHz.
* Hard Disk : 40 GB.
* Floppy Drive : 1.44 Mb.
* Monitor : 15 VGA Colour.
* Mouse : Logitech.
* Ram : 512 Mb.

**SOFTWARE REQUIREMENTS:**

* Operating system : - Windows XP.
* Coding Language : J2EE
* Data Base : MYSQL

16

## 3.5 System Architecture

## 

## Android Architecture

As can be seen in the Figure [2.1](#_bookmark11), the Android stack consists of four layers that manage the whole system starting from hardware sensors to the user’s high-level apps. Each layer provides specific services and groups of programs that run similar functions. The first layer, the Linux Kernel layer is the most important layer and located at the bottom; it represents the heart of Android system. It provides the OS services and manages the hardware’s functions such as memory, power, drivers, network stack, security set- tings, shared libraries and hardware abstraction. The second layer, the native library layer, provides native libraries which are a set of instructions that manage data processing. The native layer provides the open source libraries, such as surface manager, media framework, SQLite, Webkit, OpenGL—ES, Free Type, and SSL. In short, those libraries do the following jobs: Surface manager library is for composing windows on the screen; media framework library is for processing input and output of video and audio data; SQLite library is for database operations; Web kit is for supporting web browsers; OpenGL—ES is for supporting high performance 2D and 3D graphics; Free Type is for fonts support; SSL library is for providing services of SSL and TLS protocols. This layer also provides the Android runtime libraries which include the core libraries and the Dalvik VM. The Core libraries are a group of Java core libraries for developing An- droid apps. The Dalvik VM is the virtual environment for sandboxing the apps where they are isolated and run separately in a way for securing and optimizing resources’uses. The third layer, the Application Framework Layer, includes the Android APIs. The APIs are classes and interfaces for Android apps’ development. This layer interacts with the running apps and manages the basic functions on the device. The most important programs in this layer are activity manager, content provider, telephony manager, location manager, and resources manager. Each manager is responsible for managing a specific function. For example, the the activity manager controls life cycle of the apps. The content provider manages sharing data between the apps. The telephony manager provides sevices regarding voice calls. The resources manager regulates resources that are needed for running the apps. Finally, the Application Layer which is the topmost layer where the phone’s functions are provided to the end-user. The application layer provide functionalities that include making calls, managing contacts, sending messages, and browsing web. In this layer, Android provides a set of core applications, such as email client, calendar, browser, maps, contacts, SMS program, gallery, and etc.

17

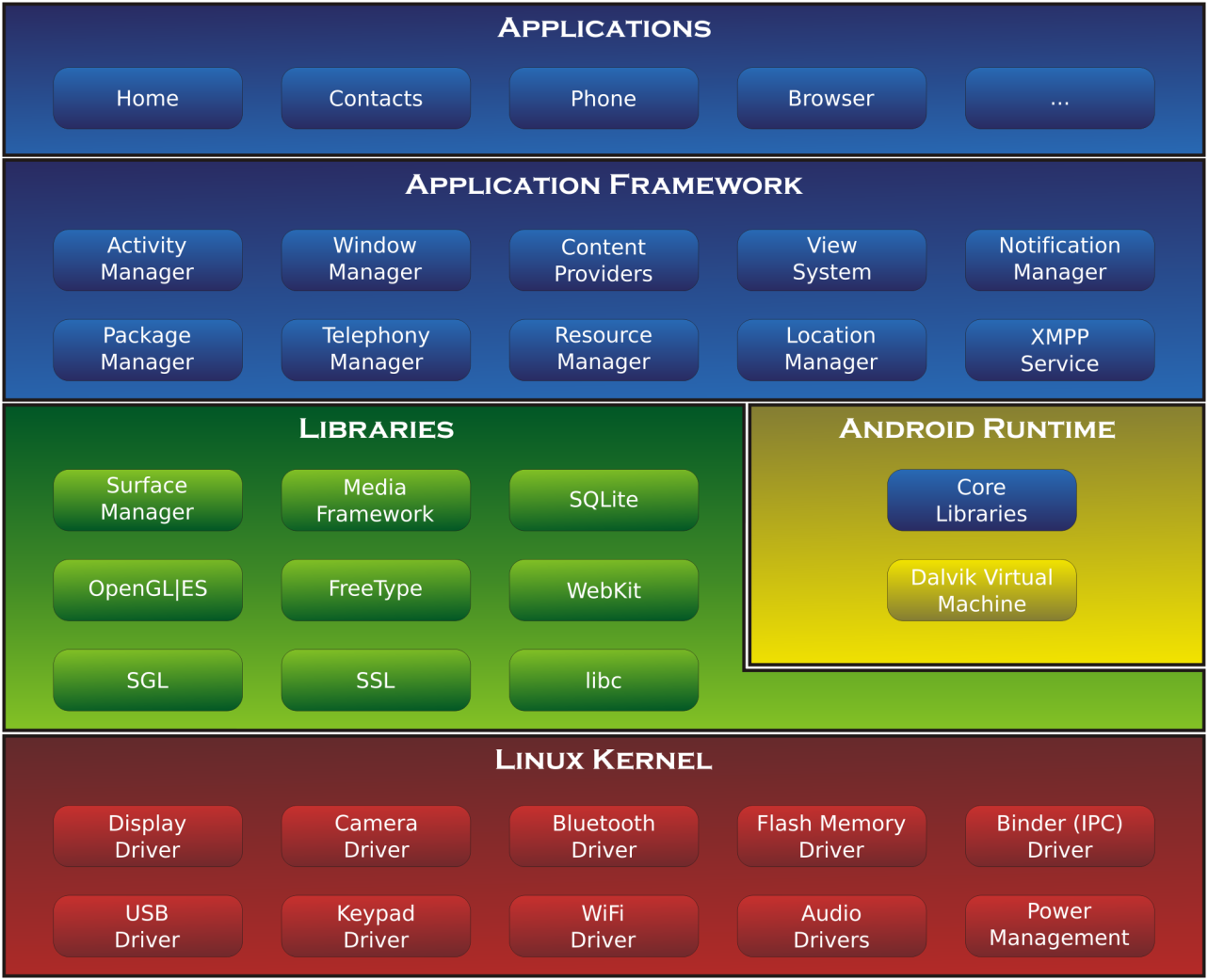


Figure 2.1: Android’s Stack Structure

## 18

## Android Malware

The table below briefly lists most common Android malware types and its characteristics

|  |  |  |
| --- | --- | --- |
| **Type** | **Definition** | **Example** |
| Trojan | Masquerades as a benign app to hide its maliciousness identity. It offers useful functionalities to the user but performs malicious activities in the background without knowledge of the user. | FakeNetFlix, Zsone,  Zitmo, Spitmo,  Fake player, Android.Foney, |
| Backdoor | Enables remote access to the system and bypasses system’s authentication mechanism. It usually exploits vulnerabilities in the system to take root’s privileges; it has ability to itself and remain undetected. | Basebridge, KMin,  Obad. |
| Worm | Copies and spreads itself over a network’s node without need to be launched by a system’s user. | Android.Obad.OS |
| Bot | Enables an attacker to remotely control the device from a server called Bot-master. The attacker commands the system other infected ones to launch an attack such as: DDoS. | Geinimi, Beanbot,  Anserverbot, |
| Spyware | Sends user’s data such as: contacts, messages,location, and other confidential data to a remote server. and activities on the device and collects | Nickyspy, GPSSpy. |
| Adware | Sends personalized advertisements based on a  user’s collected data such as location. | Plankton |
| Ransomware | Locks the system to make it inaccessible until  some ransom is paid by the user. | Fake Defender |

Table 2.1: Android Malware Families

19

**Related Work**

In this chapter, we survey some research studies in detection malware in Android environment. There are mainly two approaches are used to detect malware: static analysis and dynamic analysis. Static analysis technique examines the app’s source code with- out executing it to detect malicious patterns; the executable app is disassembled to the source code files from where many features are extracted such as: permissions, hardware components, broadcast receivers, APIs, intents, data flow, control flow, etc. On the other hand, dynamic analysis examines the app in a run time environment and monitors the app’s dynamic behavior and the system’s responses; dynamic features are monitored like network connections, system calls, resources’ usage, etc. Commonly, in both approaches, the data is collected to train machine learning classifiers to build a separation modeling between benign and malicious characteristics of the apps. The following sections show in details some research studies have been conducted in the both approaches.

20

**3.6 DATA FLOW DIAGRAM:**

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to

model the system components. These components are the system process, the data used

by the process, an external entity that interacts with the system and the information

flows in the system.

1. DFD shows how the information moves through the system and how it is modified by

a series of transformations. It is a graphical technique that depicts information flow

and the transformations that are applied as data moves from input to output.

1. DFD is also known as bubble chart. A DFD may be used to represent a system at any

level of abstraction. DFD may be partitioned into levels that represent increasing

information flow and functional detail.

21

CLASSIFICATION

TEST / TRAIN DATA

DATA PREPROCESSING

INPUT DATA SET

PREDECTION

DATA CLEANING

22

**3.7 UML Diagram**

Admin

23

Admin

24

**3.8 INPUT DESIGN**

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design

considered the following things:

What data should be given as input?

How the data should be arranged or coded?

The dialog to guide the operating personnel in providing input.

Methods for preparing input validations and steps to follow when error occur.

**OBJECTIVES**

1.Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3.When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

25

**OUTPUT DESIGN**

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2.Select methods for presenting information.

3.Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

* Convey information about past activities, current status or projections of the
* Future.
* Signal important events, opportunities, problems, or warnings.
* Trigger an action.
* Confirm an action.

26

**IMPLEMENTATION**

**4.1 Modules:-**

**(Data Collection and Feature Filtering)**

1. Collect all applications in separate folders which contain benign as well as suspicious applications respectively.
2. Using “Glob” framework in python create an array of files is for further processing.
3. Analyze each application in the array using “pyaxmlparser” and “androguard” framework.
4. Extract the following things in the analysis phase:
   1. Permissions
   2. Activities
   3. Intents
   4. API calls
5. Taking these four attributes into consideration a program maps all attributes to a CSV file and mentions a class for each application.
6. Once CSV files are generated, analyze them for any redundancy present, and if found, eliminate the entire row.
7. Another program extracts the total permissions from these APK files. These permissions will work as attributes in the Dataset CSV File (Here if permission is present it is marked as 1 else it is marked as 0).

An N-bit Vector extracts search line in the CSV file, these vectors work as input to the machine learning algorithm.

27

**4.2 Software Environment**

**4.2.1 PYTHON**

An Introduction to Python

Python is a popular object-oriented programming language having the capabilities of high-level programming language. Its easy to learn syntax and portability capability makes it popular these days. The followings facts gives us the introduction to Python −

* Python was developed by Guido van Rossum at Stitching Mathematisch Centrum in the Netherlands.
* It was written as the successor of programming language named ‘ABC’.
* It’s first version was released in 1991.
* The name Python was picked by Guido van Rossum from a TV show named Monty Python’s Flying Circus.
* It is an open source programming language which means that we can freely download it and use it to develop programs. It can be downloaded from [www.python.org.](https://www.python.org/).
* Python programming language is having the features of Java and C both. It is having the elegant ‘C’ code and on the other hand, it is having classes and objects like Java for object-oriented programming.
* It is an interpreted language, which means the source code of Python program would be first converted into byte code and then executed by Python virtual machine.

28

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). This **tutorial** gives enough understanding on **Python programming** language.

**4.2.2 Why to Learn Python?**

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

Python is a MUST for students and working professionals to become a great Software Engineer specially when they are working in Web Development Domain. I will list down some of the key advantages of learning Python:

* **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

29

**Characteristics of Python**

Following are important characteristics of **Python Programming** −

* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* It supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

**Hello World using Python.**

Just to give you a little excitement about Python, I'm going to give you a small conventional Python Hello World program.

print ("Hello, Python!");

**Applications of Python**

As mentioned before, Python is one of the most widely used language over the web. I'm going to list few of them here:

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
* **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

30

* **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases** − Python provides interfaces to all major commercial databases.
* **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable** − Python provides a better structure and support for large programs than shell scripting.

# **Python Applications**

Python is known for its general-purpose nature that makes it applicable in almost each domain of software development. Python as a whole can be used in any sphere of development.

Here, we are specifying applications areas where python can be applied.

1. Web Applications

We can use Python to develop web applications. It provides libraries to handle internet protocols such as HTML and XML, JSON, Email processing, request, beautiful Soup, Feed parser etc. It also provides Frameworks such as Django, Pyramid, Fletch to design and develop web based application .

31

Some important developments are: Python Wiki Engines, Pocoo, Python Blog Software etc.

#### 2) Desktop GUI Applications

Python provides Tk GUI library to develop user interface in python based application. Some other useful toolkits wxWidgets, Kivy, pyqt that are useable on several platforms. The Kivy is popular for writing multi touch applications.

#### 3) Software Development

Python is helpful for software development process. It works as a support language and can be used for build control and management, testing etc.

#### 4) Scientific and Numeric

Python is popular and widely used in scientific and numeric computing. Some useful library and package are SciPy, Pandas, IPython etc. SciPy is group of packages of engineering, science and mathematics.

#### 5) Business Applications

Python is used to build Business applications like ERP and e-commerce systems. Tryton is a high level application platform.

#### 6) Console Based Application

We can use Python to develop console based applications. For example: **IPython**.

#### 7) Audio or Video based Applications

Python is awesome to perform multiple tasks and can be used to develop multimedia applications. Some of real applications are: TimPlayer, cplay etc.

#### 32

#### 8) 3D CAD Applications

To create CAD application Fandango is a real application which provides full features of CAD.

#### 9) Enterprise Applications

Python can be used to create applications which can be used within an Enterprise or an Organization. Some real time applications are: Open Erp, Tryton, Picalo etc.

#### 10) Applications for Images

Using Python several application can be developed for image. Applications developed are: VPython, Gogh, img Seek etc.

There are several such applications which can be developed using Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

* **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

## 33

## History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Small Talk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

## Python Features

Python's features include −

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − Python's source code is fairly easy-to-maintaining.
* **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

34

* **Databases** − Python provides interfaces to all major commercial databases.
* **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* It supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

## Local Environment Setup

Open a terminal window and type "python" to find out if it is already installed and which version is installed.

* Unix (Solaris, Linux, FreeBSD, AIX, HP/UX, SunOS, IRIX, etc.)
* Win 9x/NT/2000
* Macintosh (Intel, PPC, 68K)

35

* OS/2
* DOS (multiple versions)
* Palm OS
* Nokia mobile phones
* Windows CE
* Acorn/RISC OS
* BeOS
* Amiga
* VMS/OpenVMS
* QNX
* Vx Works
* Psion
* Python has also been ported to the Java and .NET virtual machines

## Getting Python

The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python <https://www.python.org/>

You can download Python documentation from <https://www.python.org/doc/>. The documentation is available in HTML, PDF, and PostScript formats.

## 36

## Installing Python

Python distribution is available for a wide variety of platforms. You need to download only the binary code applicable for your platform and install Python.

If the binary code for your platform is not available, you need a C compiler to compile the source code manually. Compiling the source code offers more flexibility in terms of choice of features that you require in your installation.

Here is a quick overview of installing Python on various platforms −

### Unix and Linux Installation

Here are the simple steps to install Python on Unix/Linux machine.

* Open a Web browser and go to <https://www.python.org/downloads/>.
* Follow the link to download zipped source code available for Unix/Linux.
* Download and extract files.
* Editing the Modules/Setup file if you want to customize some options.
* run ./configure script
* make
* make install

This installs Python at standard location /usr/local/bin and its libraries

at /usr/local/lib/pythonXX where XX is the version of Python.

37

### Here are the steps to install Python on Windows machine.

* Open a Web browser and go to <https://www.python.org/downloads/>.
* Follow the link for the Windows installer python-XYZ.msi file where XYZ is the version you need to install.
* To use this installer python-XYZ.msi, the Windows system must support Microsoft Installer 2.0. Save the installer file to your local machine and then run it to find out if your machine supports MSI.
* Run the downloaded file. This brings up the Python install wizard, which is really easy to use. Just accept the default settings, wait until the install is finished, and you are done.

### Macintosh Installation

Recent Macs come with Python installed, but it may be several years out of date. See [http://www.python.org/download/mac/](https://www.python.org/download/mac/) for instructions on getting the current version along with extra tools to support development on the Mac. For older Mac OS's before Mac OS X 10.3 (released in 2003), Mac Python is available.

Jack Jansen maintains it and you can have full access to the entire documentation at his website − <http://www.cwi.nl/~jack/macpython.html>. You can find complete installation details for Mac OS installation.

## 

## 38

## Setting up PATH

Programs and other executable files can be in many directories, so operating systems provide a search path that lists the directories that the OS searches for executables.

The path is stored in an environment variable, which is a named string maintained by the operating system. This variable contains information available to the command shell and other programs.

The **path** variable is named as PATH in Unix or Path in Windows (Unix is case sensitive; Windows is not).

In Mac OS, the installer handles the path details. To invoke the Python interpreter from any particular directory, you must add the Python directory to your path.

## Setting path at Unix/Linux

To add the Python directory to the path for a particular session in Unix −

* **In the csh shell** − type setenv PATH "$PATH:/usr/local/bin/python" and press Enter.
* **In the bash shell (Linux)** − type export PATH="$PATH:/usr/local/bin/python" and press Enter.
* **In the sh or ksh shell** − type PATH="$PATH:/usr/local/bin/python" and press Enter.
* **Note** − /usr/local/bin/python is the path of the Python directory

## Setting path at Windows

To add the Python directory to the path for a particular session in Windows −

**At the command prompt** − type path %path%; C:\Python and press Enter.

**Note** − C:\Python is the path of the Python directory.

39

**Python with Deep Learning**

# **Introduction to Deep Learning**

**What is Deep Learning?**

Deep learning is a branch of [machine learning](https://www.geeksforgeeks.org/introduction-machine-learning/) which is completely based on [artificial neural networks](https://www.geeksforgeeks.org/tag/neural-network/), as neural network is going to mimic the human brain so deep learning is also a kind of mimic of human brain. In deep learning, we don’t need to explicitly program everything. The concept of deep learning is not new. It has been around for a couple of years now. It’s on hype nowadays because earlier we did not have that much processing power and a lot of data. As in the last 20 years, the processing power increases exponentially, deep learning and machine learning came in the picture.  
A formal definition of deep learning is- neurons.

Deep learning is a particular kind of machine learning that achieves great power and flexibility by learning to represent the world as a nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones.

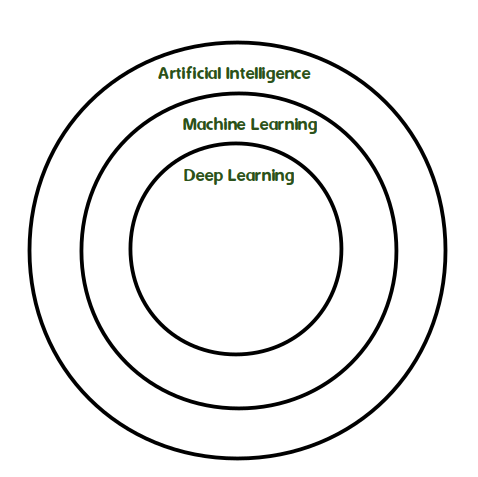
In human brain approximately 100 billion neurons all together this is a picture of an individual neuron and each neuron is connected through thousand of their neighbours.  
The question here is how do we recreate these neurons in a computer. So, we create an artificial structure called an artificial neural net where we have nodes or neurons. We have some neurons for input value and some for output value and in between, there may be lots of neurons interconnected in the hidden layer.

**Architectures :**

1. **Deep Neural Network** – It is a neural network with a certain level of complexity (having multiple hidden layers in between input and output layers). They are capable of modeling and processing non-linear relationships.

**40**

1. **Deep Belief Network(DBN)** – It is a class of Deep Neural Network. It is multi-layer belief networks.  
   **Steps for performing DBN :**  
   a. Learn a layer of features from visible units using Contrastive Divergence algorithm.  
   b. Treat activations of previously trained features as visible units and then learn features of features.  
   c. Finally, the whole DBN is trained when the learning for the final hidden layer is achieved.
2. **Recurrent** (perform same task for every element of a sequence)**Neural Network** – Allows for parallel and sequential computation. Similar to the human brain (large feedback network of connected neurons). They are able to remember important things about the input they received and hence enables them to be more precise.



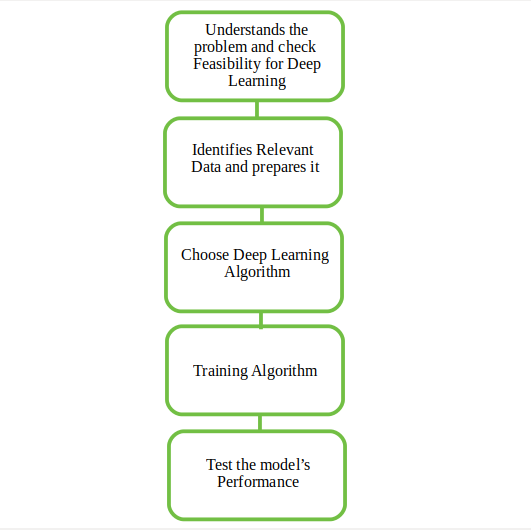
**41**

**Difference between Machine Learning and Deep Learning :**

| **MACHINE LEARNING** | **DEEP LEARNING** |
| --- | --- |
| Works on small amount of Dataset for accuracy. | Works on Large amount of Dataset. |
| Dependent on Low-end Machine. | Heavily dependent on High-end Machine. |
| Divides the tasks into sub-tasks, solves them individually and finally combine the results. | Solves problem end to end. |
| Takes less time to train. | Takes longer time to train. |
| Testing time may increase. | Less time to test the data. |

**Working:**  
First, we need to identify the actual problem in order to get the right solution and it should be understood, the feasibility of the Deep Learning should also be checked (whether it should fit Deep Learning or not). Second, we need to identify the relevant data which should correspond to the actual problem and should be prepared accordingly.

Third, Choose the Deep Learning Algorithm appropriately. Fourth, Algorithm should be used while training the dataset. Fifth, Final testing should be done on the dataset.

42  


**Tools used :**  
Anaconda, Jupyter, Pycharm, etc.

**Languages used :**

R, Python, Matlab, CPP, Java, Julia, Lisp, Java Script, etc.

**Real Life Examples :**

1. How to recognize square from other shapes?
2. ...a) Check the four lines!
3. ...b) Is it a closed figure?
4. ...c) Does the sides are perpendicular from each other?
5. ...d) Does all sides are equal?
6. So, Deep Learning is a complex task of identifying the shape and broken down into simpler
7. tasks at a larger side.
8. Recognizing an Animal! (Is it a Cat or Dog?)

43

1. Defining facial features which are important for classification and system will then identify this automatically.
2. (Whereas Machine Learning will manually give out those features for classification)

**Benefits or advantages of Deep Learning**

Following are the benefits or advantages of Deep Learning:

➨Features are automatically deduced and optimally tuned for desired outcome. Features are not required to be extracted ahead of time. This avoids time consuming machine learning techniques.

➨Robustness to natural variations in the data is automatically learned.

➨The same neural network based approach can be applied to many different applications and data types.

➨Massive parallel computations can be performed using GPUs and are scalable for large volumes of data. Moreover it delivers better performance results when amount of data are huge.

➨The deep learning architecture is flexible to be adapted to new problems in the future.

**Drawbacks or disadvantages of Deep Learning**

Following are the drawbacks or disadvantages of Deep Learning:

➨It requires very large amount of data in order to perform better than other techniques.

➨It is extremely expensive to train due to complex data models. Moreover deep learning requires expensive GPUs and hundreds of machines. This increases cost to the users.

➨There is no standard theory to guide you in selecting right deep learning tools as it requires knowledge of topology, training method and other parameters. As a result it is difficult to be adopted by less skilled people.

➨It is not easy to comprehend output based on mere learning and requires classifiers to do so. Convolution neural network based algorithms perform such tasks.

44

# **Machine Learning with Python**

# 

Machine Learning is simply making a computer perform a task without explicitly programming it. In today’s world every system that does well has a machine learning algorithm at its heart. Take for example Google Search engine, Amazon Product recommendations, LinkedIn, Facebook etc, all these systems have machine learning algorithms embedded in their systems in one form or the other. They are efficiently utilising data collected from various channels which helps them get a bigger picture of what they are doing and what they should do.

Python is a widely used high-level programming language for general-purpose programming. Apart from being open source programming language, python is a great object-oriented, interpreted, and interactive programming language. Python combines remarkable power with very clear syntax. It has modules, classes, exceptions, very high level dynamic data types, and dynamic typing. There are interfaces to many system calls and libraries, as well as to various windowing systems. New built-in modules are easily written in C or C++ (or other languages, depending on the chosen implementation). Python is also usable as an extension language for applications written in other languages that need easy-to-use scripting or automation interfaces.

45

Machine Learning (ML) is basically that field of computer science with the help of which computer systems can provide sense to data in much the same way as human beings do. In simple words, ML is a type of artificial intelligence that extract patterns out of raw data by using an algorithm or method. The key focus of ML is to allow computer systems to learn from experience without being explicitly programmed or human intervention.

We are living in the ‘age of data’ that is enriched with better computational power and more storage resources,. This data or information is increasing day by day, but the real challenge is to make sense of all the data. Businesses & organizations are trying to deal with it by building intelligent systems using the concepts and methodologies from Data science, Data Mining and Machine learning. Among them, machine learning is the most exciting field of computer science. It would not be wrong if we call machine learning the application and science of algorithms that provides sense to the data.

## What is Machine Learning?

Machine Learning (ML) is that field of computer science with the help of which computer systems can provide sense to data in much the same way as human beings do.

In simple words, ML is a type of artificial intelligence that extract patterns out of raw data by using an algorithm or method. The main focus of ML is to allow computer systems learn from experience without being explicitly programmed or human intervention.

## Need for Machine Learning

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate and solve complex problems. On the other side, AI is still in its initial stage and haven’t surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, “to make decisions, based on data, with efficiency and scale. Lately, organizations are investing heavily in newer technologies like Artificial Intelligence

46

Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can be used, instead of using programing logic, in the problems that cannot be programmed inherently. The fact is that we can’t do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

## Why & When to Make Machines Learn?

We have already discussed the need for machine learning, but another question arises that in what scenarios we must make the machine learn? There can be several circumstances where we need machines to take data-driven decisions with efficiency and at a huge scale. The followings are some of such circumstances where making machines learn would be more effective.

### Lack of human expertise

The very first scenario in which we want a machine to learn and take data-driven decisions, can be the domain where there is a lack of human expertise. The examples can be navigations in unknown territories or spatial planets.

### Dynamic scenarios

There are some scenarios which are dynamic in nature i.e. they keep changing over time. In case of these scenarios and behaviors, we want a machine to learn and take data-driven decisions. Some of the examples can be network connectivity and availability of infrastructure in an organization.

### Difficulty in translating expertise into computational tasks

There can be various domains in which humans have their expertise,; however, they are unable to translate this expertise into computational tasks. In such circumstances we want machine learning.

47

**Machine Learning Model**

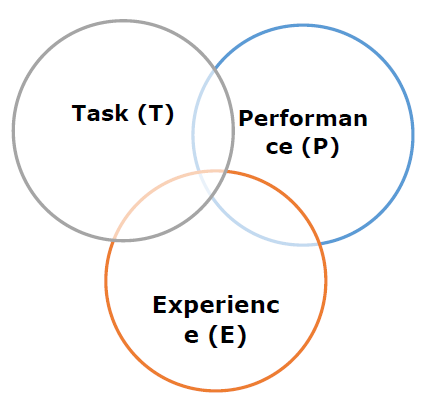
Before discussing the machine learning model, we must need to understand the following formal definition of ML given by professor Mitchell −

“A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.”

The above definition is basically focusing on three parameters, also the main components of any learning algorithm, namely Task(T), Performance(P) and experience (E). In this context, we can simplify this definition as −

**ML is a field of AI consisting of learning algorithms that −**

* Improve their performance (P)
* At executing some task (T)
* Over time with experience (E)

Based on the above, the following diagram represents a Machine Learning Model 

48

Let us discuss them more in detail now −

### Task(T)

From the perspective of problem, we may define the task T as the real-world problem to be solved. The problem can be anything like finding best house price in a specific location or to find best marketing strategy etc. On the other hand, if we talk about machine learning, the definition of task is different because it is difficult to solve ML based tasks by conventional programming approach.A task T is said to be a ML based task when it is based on the process and the system must follow for operating on data points. The examples of ML based tasks are Classification, Regression, Structured annotation, Clustering, Transcription etc.

### Experience (E)

As name suggests, it is the knowledge gained from data points provided to the algorithm or model. Once provided with the dataset, the model will run iteratively and will learn some inherent pattern. The learning thus acquired is called experience(E). Making an analogy with human learning, we can think of this situation as in which a human being is learning or gaining some experience from various attributes like situation, relationships etc.

Supervised, unsupervised and reinforcement learning are some ways to learn or gain experience. The experience gained by out ML model or algorithm will be used to solve the task T.

### Performance (P)

An ML algorithm is supposed to perform task and gain experience with the passage of time. The measure which tells whether ML algorithm is performing as per expectation or not is its performance (P). P is basically a quantitative metric that tells how a model is performing the task, T, using its experience, E. There are many metrics that help to understand the ML performance, such as accuracy score, F1 score, confusion matrix, precision, recall, sensitivity etc.

49

**Challenges in Machines Learning**

While Machine Learning is rapidly evolving, making significant strides with cyber security and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are −

**Quality of data** − Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.

**Time-Consuming task** − Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.

**Lack of specialist persons** − As ML technology is still in its infancy stage, availability of expert resources is a tough job.

**No clear objective for formulating business problems** − Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

**Issue of overfitting & underfitting** − If the model is over fitting or under fitting, it cannot be represented well for the problem.

**Curse of dimensionality** − Another challenge ML model faces is too many features of data points. This can be a real hindrance.

**Difficulty in deployment** − Complexity of the ML model makes it quite difficult to be deployed in real life.

50

**Applications of Machines Learning**

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML

* Emotion analysis
* Sentiment analysis
* Error detection and prevention
* Weather forecasting and prediction
* Stock market analysis and forecasting
* Speech synthesis
* Speech recognition
* Customer segmentation
* Object recognition
* Fraud detection
* Fraud prevention
* Recommendation of products to customer in online shopping

## Components of Python ML Ecosystem

In this section, let us discuss some core Data Science libraries that form the components of Python Machine learning ecosystem. These useful components make Python an important language for Data Science.

Though there are many such components, let us discuss some of the importance components of Python ecosystem here −

* [Jupyter Notebook](https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_jupyter_notebook.htm) − Jupyter notebooks basically provides an interactive computational environment for developing Python based Data Science applications.

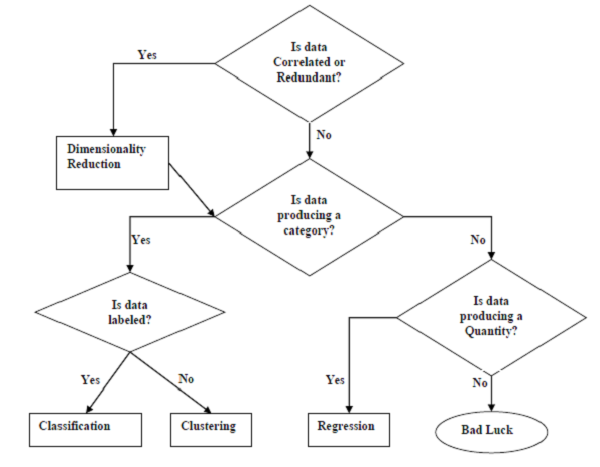
## 51

## Different Types of Methods

The following are various ML methods based on some broad categories −

* [Based on human supervision](https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_based_on_human_supervision.htm)
* [Unsupervised Learning](https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_unsupervised_learning.htm)
* [Semi-supervised Learning](https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_semi_supervised_learning.htm)
* [Reinforcement Learning](https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_reinforcement_learning.htm)

## Tasks Suited for Machine Learning

The following diagram shows what type of task is appropriate for various ML problems

52

### Based on learning ability

In the learning process, the following are some methods that are based on learning ability

**Batch Learning**

In many cases, we have end-to-end Machine Learning systems in which we need to train the model in one go by using whole available training data. Such kind of learning method or algorithm is called **Batch or Offline learning**. It is called Batch or Offline learning because it is a one-time procedure and the model will be trained with data in one single batch. The following are the main steps of Batch learning methods

**Step 1** − First, we need to collect all the training data for start training the model.

* **Step 2** − Now, start the training of model by providing whole training data in one go.
* **Step 3** − Next, stop learning/training process once you got satisfactory results/performance.
* **Step 4** − Finally, deploy this trained model into production. Here, it will predict the output for new data sample.

### Online Learning

It is completely opposite to the batch or offline learning methods. In these learning methods, the training data is supplied in multiple incremental batches, called mini-batches, to the algorithm. Followings are the main steps of Online learning methods −

* **Step 1** − First, we need to collect all the training data for starting training of the model.
* **Step 2** − Now, start the training of model by providing a mini-batch of training data to the algorithm.
* **Step 3** − Next, we need to provide the mini-batches of training data in multiple increments to the algorithm.

53

* **Step 4** − As it will not stop like batch learning hence after providing whole training data in mini-batches, provide new data samples also to it.
* **Step 5** − Finally, it will keep learning over a period of time based on the new data samples.

### Based on Generalization Approach

In the learning process, followings are some methods that are based on generalization approaches −

### Instance based Learning

Instance based learning method is one of the useful methods that build the ML models by doing generalization based on the input data. It is opposite to the previously studied learning methods in the way that this kind of learning involves ML systems as well as methods that uses the raw data points themselves to draw the outcomes for newer data samples without building an explicit model on training data.

In simple words, instance-based learning basically starts working by looking at the input data points and then using a similarity metric, it will generalize and predict the new data points.

### Model based Learning

In Model based learning methods, an iterative process takes place on the ML models that are built based on various model parameters, called hyper parameters and in which input data is used to extract the features. In this learning, hyper parameters are optimized based on various model validation techniques. That is why we can say that Model based learning methods uses more traditional ML approach towards generalization

54

### Python for Machine Language (ML)

Let us look as to why Python is used for Machine Learning and the various libraries it offers for the purpose.

* **PyBrain** -  A flexible, simple yet effective algorithm for ML tasks. It is also a modular Machine Learning Library for Python providing a variety of predefined environments to test and compare algorithms.
* **PyML** – A bilateral framework written in Python that focuses on SVMs and other kernel methods. It is supported on Linux and Mac OS X.
* **Scikit-learn** – Scikit-learn is an efficient tool for data analysis while using Python. It is open source and the most popular general purpose machine learning library.
* **MDP-Toolkit** – Another Python data processing framework that can be easily expanded, it also has a collection of supervised and unsupervised learning algorithms and other data processing units that can be combined into data processing sequences and more complex feed-forward network architectures. The implementation of new algorithms is easy and intuitive. The base of available algorithms is steadily increasing and includes signal processing methods (Principal Component Analysis, Independent Component Analysis, and Slow Feature Analysis), manifold learning methods ([Hessian] Locally Linear Embedding), several classifiers, probabilistic methods (Factor Analysis, RBM), data pre-processing methods, and many others.

55

**What are the advantages and disadvantages of machine learning?**

**Advantages Of Machine Learning**

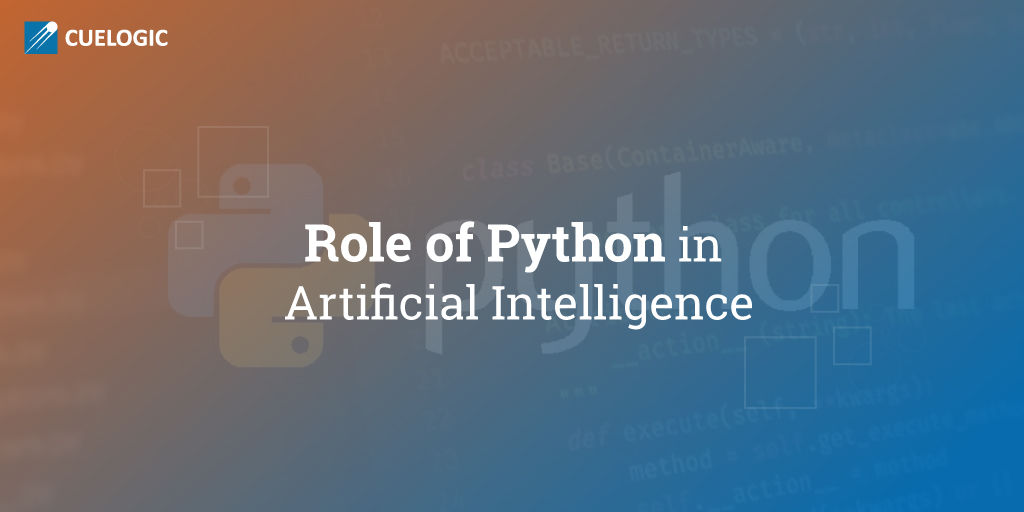
1. Identifies trends and patterns easily- Machine learning involves reviewing large volumes of data to discover specific trends and patterns that would most often not be apparent to humans. For example, machine learning will be useful for an e-commerce website like Amazon, to understand purchase histories and browsing behaviours of its users to cater to the right deals, products, and reminders that are relevant to them.
2. Automation- Machine learning does not require human intervention. It gives machines the ability to learn. It helps machines make predictions and improve the algorithms by themselves. Anti-virus software is a common example of this as they automatically filter new threats as & when they are recognized.
3. Continuous improvement- Machine learning algorithms improve in accuracy and efficiency as they gain experience. This helps them take better decisions.

**Disadvantages Of Machine Learning**

1. Time and resources- Machine learning requires massive resources to function. It may demand additional computing power. Machine learning requires enough time to let the algorithms learn & develop to fulfill their intended purpose with a considerable amount of accuracy and relevancy.
2. Interpretation of results- Accurately interpreting the results generated by the algorithms is a challenging task. One needs to exercise caution while choosing algorithms for their specific purpose.
3. Data acquisition- Machine learning needs massive datasets to train on. These must be unbiased/inclusive and of good quality. In certain situations, they may need to wait for new data to be generate.

56

**Python in Artificial Intelligence (AI)**



## Python and Artificial Intelligence(AI) - How do they relate?

Python is one of the most popular programming languages used by developers today. Guido Van Rossum created it in 1991 and ever since its inception has been one of the most widely used languages along with C++, Java, etc.

In our Endeavour to identify what is the best [programming language for AI](https://www.cuelogic.com/blog/python-in-finance-analytics-artificial-intelligence) and neural network, Python has taken a big lead. Let us look at why [Artificial Intelligence](https://www.cuelogic.com/artificial-intelligence-solutions) with Python is one of the best ideas under the sun.

## Features and Advantages of Python

Python is an Interpreted language which in lay man’s terms means that it does not need to be compiled into machine language instruction before execution and can be used by the developer directly to run the program. This makes it comprehensive enough for the language to be interpreted by an emulator or a virtual machine on top of the native machine language which is what the hardware understands.

57

It is a [High-Level Programming](https://www.cuelogic.com/blog/10-best-iot-programming-languages) language and can be used for complicated scenarios. High-level languages deal with variables, arrays, objects, complex arithmetic or Boolean expressions, and other abstract computer science concepts to make it more comprehensive thereby exponentially increasing its usability.

Python is also a General-purpose programming language which means it can be used across domains and technologies.

Python also features dynamic type system and automatic memory management supporting a wide variety of programming paradigms including object-oriented, imperative, functional and procedural to name a few.

Python is available for all Operating Systems and also has an open-source offering titled CPython which is garnering widespread popularity as well.

Let us now look as to how using Python for Artificial Inelegance gives us an edge over other popular programming languages.

## AI and Python: Why?

The obvious question that we need to encounter at this point is why we should choose [Python for AI over others](https://www.cuelogic.com/blog/advanced-image-processing-with-python).

Python offers the least code among others and is in fact 1/5 the number compared to other OOP languages. No wonder it is one of the most popular in the market today.

* Python has Prebuilt Libraries like Numpy for scientific computation, Scipy for advanced computing and Pybrain for machine learning (Python Machine Learning) making it one of the best languages For AI.
* Python developers around the world provide comprehensive support and assistance via forums and tutorials making the job of the coder easier than any other popular languages 58
* Python is platform Independent and is hence one of the most flexible and popular choice for use across different platforms and technologies with the least tweaks in basic coding.
* Python is the most flexible of all others with options to choose between OOPs approach and scripting. You can also use IDE itself to check for most codes and is a boon for developers struggling with different algorithms.

## Decoding Python alongside AI

Python along with packages like NumPy, scikit-learn, iPython Notebook, and matplotlib form the basis to start your AI project.

NumPy is used as a container for generic data comprising of an N-dimensional array object, tools for integrating C/C++ code, Fourier transform, random number capabilities, and other functions.

Another useful library is pandas, an open source library that provides users with easy-to-use data structures and analytic tools for Python.

Matplotlib is another service which is a 2D plotting library creating publication quality figures. You can use matplotlib to up to 6 graphical users interface toolkits, web application servers, and Python scripts.

Your next step will be to explore k-means clustering and also gather knowledge about decision trees, continuous numeric prediction, logistic regression, etc.

Some of the most commonly used Python AI libraries are AIMA, pyDatalog, SimpleAI, EasyAi, etc. There are also Python libraries for machine learning like PyBrain, MDP, scikit, PyML.Let us look a little more in detail about the various Python libraries in AI and why this programming language is used for AI.

59

### Python Libraries for General AI

* **AIMA** – Python implementation of algorithms from Russell and Norvig’s ‘Artificial Intelligence: A Modern Approach.’
* **pyDatalog** – Logic Programming engine in Python
* **SimpleAI** – Python implementation of many of the artificial intelligence algorithms described on the book “Artificial Intelligence, a Modern Approach”. It focuses on providing an easy to use, well documented and tested library.
* **EasyAI** – Simple Python engine for two-players games with AI (Negamax, transposition tables, game solving).

**Advantages of Artificial intelligence :**

1. **Reduction in Human Error:** The phrase “**human error**” was born because humans make mistakes from time to time. Computers, however, do not make these mistakes if they are programmed properly. With Artificial intelligence, the decisions are taken from the previously gathered information applying a certain set of algorithms. So errors are reduced and the chance of reaching accuracy with a greater degree of precision is a possibility.

**Example:**In Weather Forecasting using AI they have reduced the majority of human error.

**2) Takes risks instead of Humans:**

This is one of the biggest advantages of Artificial intelligence. We can overcome many risky limitations of humans by developing an AI Robot which in turn can do the risky things for us. Let it be going to mars, defuse a bomb, explore the deepest parts of oceans, mining for coal and oil, it can be used effectively in any kind of natural or man-made disasters.

**60**

**Example:**Have you heard about the **Chernoby**l nuclear power plant explosion in Ukraine? At that time there were no AI-powered robots that can help us to minimize the effect of radiation by controlling the fire in early stages, as any human went close to the core was dead in a matter of minutes. They eventually poured sand and boron from helicopters from a mere distance.AI Robots can be used in such situations where intervention can be hazardous.

**3) Available 24x7:**

An Average human will work for 4–6 hours a day excluding the breaks. Humans are built in such a way to get some time out for refreshing themselves and get ready for a new day of work and they even have weekly offed to stay intact with their work-life and personal life. But using AI we can make machines work 24x7 without any breaks and they don’t even get bored, unlike humans.

**Example:**Educational Institutes and Helpline centers are getting many queries and issues which can be handled effectively using AI.

**4) Helping in Repetitive Jobs:**

In our day-to-day work, we will be performing many repetitive works like sending a thanking mail, verifying certain documents for errors and many more things. Using artificial intelligence we can productively automate these mundane tasks and can even remove “**boring**” tasks for humans and free them up to be increasingly creative.

**Example:**In banks, we often see many verifications of documents to get a loan which is a repetitive task for the owner of the bank. Using AI Cognitive Automation the owner can speed up the process of verifying the documents by which both the customers and the owner will be benefited. 61

**5) Digital Assistance:**

Some of the highly advanced organizations use digital assistants to interact with users which saves the need for human resources. The digital assistants also used in many websites to provide things that users want. We can chat with them about what we are looking for. Some chat bots are designed in such a way that it’s become hard to determine that we’re chatting with a chat bots or a human being.

**Example:**We all know that organizations have a customer support team that needs to clarify the doubts and queries of the customers. Using AI the organizations can set up a Voice bots or Chat bots

which can help customers with all their queries. We can see many organizations already started using them on their websites and mobile applications.

**6) Faster Decisions:**

Using AI alongside other technologies we can make machines take decisions faster than a human and carry out actions quicker. While taking a decision human will analyze many factors both emotionally and practically but AI-powered machine works on what it is programmed and delivers the results in a faster way.

**Example:**We all have played Chess games in Windows. It is nearly impossible to beat CPU in the hard mode because of the AI behind that game. It will take the best possible step in a very short time according to the algorithms used behind it

**62**

**7) Daily Applications:**

Daily applications such as Apple’s **Siri**, Window’s **Cortana**, Google’s **OK Google** are frequently used in our daily routine whether it is for searching a location, taking a selfie, making a phone call, replying to a mail and many more.

**Example:**Around 20 years ago, when we are planning to go somewhere we used to ask a person who already went there for the directions. But now all we have to do is say “**OK Google**where is Visakhapatnam”. It will show you Visakhapatnam’s location on goggle map and the best path between you and Visakhapatnam.

**8) New Inventions:**

AI is powering many inventions in almost every domain which will help humans solve the majority of complex problems.

**Example:**Recently doctors can predict breast cancer in the woman at earlier stages using advanced AI-based technologies.

**Artificial Intelligence also has some disadvantages:**

**1) High Costs of Creation:**

As AI is updating every day the hardware and software need to get updated with time to meet the latest requirements. Machines need repairing and maintenance which need plenty of costs. It’ s creation requires huge costs as they are very complex machines.

**63**

**2) Making Humans Lazy:**

AI is making humans lazy with its applications automating the majority of the work. Humans tend to get **addicted**to these inventions which can cause a problem to future generations.

**3) Unemployment:**

As AI is replacing the majority of the repetitive tasks and other works with robots, human interference is becoming less which will cause a major problem in the employment standards. Every organization is looking to replace the minimum qualified individuals with AI robots which can do similar work with more efficiency.

**4) No Emotions:**

There is no doubt that machines are much better when it comes to working efficiently but they cannot replace the human connection that makes the team. Machines cannot develop a bond with humans which is an essential attribute when comes to Team Management.

**5) Lacking Out of Box Thinking:**

Machines can perform only those tasks which they are designed or programmed to do, anything out of that they tend to crash or give irrelevant outputs which could be a major backdrop.

64

**4.3 Sample Code**

**3.1 Support Vector Machines**

Support Vector Machines (SVM) and also called Support Vector Networks (SVN) are supervised machine learning methods to analyze, detect and match patterns of data for classification and regression purposes. SVM is a non-probabilistic binary linear classifier that assigns training data into one category or more. It also can be used efficiently for nonlinear classification problems using Kernel Trick. Kernel Trick is a class of SVM algorithms that maps the input features into a very high dimensional output space in a simple and cheaper computational way. SVM is a representation of training data as points in the space that conglomerate based on their category in form of groups that are separated by a clear distinct gap called a hyper plane. In the training phase, SVM builds up a model of patterns from the training data which is used as a space for classification phase. In the classification phase, the new input points are mapped into the trained space and categorized based on which side of the gap they fall on. In the figure below a straight line separates between two classes, the new data are mapped into the space if they up the line will be categorized into otherwise into. Hyper plane is a subspace less by one dimension than its ambient space; it is 2-dimensions in 3D space, and a 1-dimension in 2D space. SVM forms a hyper plane or set of hyper planes for data classification and regression. For more confidence and less generalization error, the hyper plane must be selected by functional margin that makes the distance between the nearest training data points in any class as much larger as possible. We selected SVM because its resistance to over-fitting even in the very high dimensional variables space like our datasets. SVM is the best choice for binary classification problems like ours.

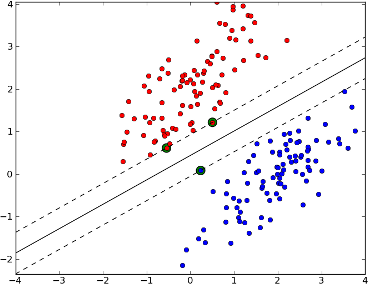


Figure 4.17: Support Vector Machines

65

**4.3.2 Main Program**

{

"cells": [

{

"cell\_type": "code",

"execution\_count": 0,

"metadata": {

"colab": {},

"colab\_type": "code",

"id": "d62XgEmMFey3"

},

"outputs": [],

"source": []

},

{

"cell\_type": "markdown",

"metadata": {

"colab\_type": "text",

"id": "L2D554VlFsiS"

},

"source": [

"Dataset Creation"

]

},

{

"cell\_type": "code",

"execution\_count": 0,

"metadata": {

"cellView": "both",

"colab": {},

"colab\_type": "code",

"id": "eWbwrUgYFzPk"

},

"outputs": [],

"source": [

"import os\n",

"\n",

"\n",

"# find the dataset here: https://www.unb.ca/cic/datasets/invesandmal2019.html\n",

"!wget http://205.174.165.80/CICDataset/CICInvesAndMal2019/Dataset/APKs/Adware.zip\n",

"!wget http://205.174.165.80/CICDataset/CICInvesAndMal2019/Dataset/APKs/PremiumSMS.zip\n",

"!wget http://205.174.165.80/CICDataset/CICInvesAndMal2019/Dataset/APKs/Ransomware.zip\n",

"!wget http://205.174.165.80/CICDataset/CICInvesAndMal2019/Dataset/APKs/SMS.zip\n",

"!wget http://205.174.165.80/CICDataset/CICInvesAndMal2019/Dataset/APKs/Scareware.zip\n",

"\n",

"#!wget http:........Benign\_2015.zip\n",

"#!wget http:........Benign\_2016.zip\n",

"#!wget http:........Benign\_2017.zip\n",

"\n",

66

"!unzip /content/Adware.zip\n",

"os.remove('/content/Adware.zip')\n",

"!unzip /content/PremiumSMS.zip\n",

"os.remove('/content/PremiumSMS.zip')\n",

"!unzip /content/SMS.zip\n",

"os.remove('/content/SMS.zip')\n",

"!unzip /content/Ransomware.zip\n",

"os.remove('/content/Ransomware.zip')\n",

"!unzip /content/Scareware.zip\n",

"os.remove('/content/Scareware.zip')\n",

"\n",

"!unzip /content/Benign\_2015.zip\n",

"os.remove('/content/Benign\_2015.zip')\n",

"!unzip /content/Benign\_2016.zip\n",

"os.remove('/content/Benign\_2016.zip')\n",

"!unzip /content/Benign\_2017.zip\n",

"os.remove('/content/Benign\_2017.zip')"

]

},

{

"cell\_type": "code",

"execution\_count": 0,

"metadata": {

"colab": {},

"colab\_type": "code",

"id": "dGT2UQTKdS\_I"

},

"outputs": [],

"source": [

"import shutil\n",

"\n",

"#!git clone https://github.com/ethicalhackeragnidhra/Android-Malwares.git /content/mal\n",

"!git clone https://github.com/RyzenElstra/Android-Malwares.git /content/mal\n",

"#!git clone https://github.com/ashishb/android-malware.git /content/mal"

]

},

{

"cell\_type": "code",

"execution\_count": 0,

"metadata": {

"colab": {},

"colab\_type": "code",

"id": "OLlDSH6d7vez"

},

"outputs": [],

"source": [

"#import shutil\n",

"#shutil.rmtree('/content/mal')"

]

},

{

67

"cell\_type": "code",

"execution\_count": 0,

"metadata": {

"colab": {},

"colab\_type": "code",

"id": "5FRe3R7jbqZ9"

},

"outputs": [],

"source": [

"\n",

"#!unzip -P 'infected' -r /content/mal"

]

},

{

"cell\_type": "code",

"execution\_count": 0,

"metadata": {

"colab": {},

"colab\_type": "code",

"id": "HU2Pd9GHQsDA"

},

"outputs": [],

"source": [

"import zipfile\n",

"\n",

"# pswd = 'infected'\n",

"\n",

"for root, dirs, files in os.walk('/content/mal'):\n",

" for file in files:\n",

" if file.endswith('.zip'):\n",

" try:\n",

" with zipfile.ZipFile(os.path.join(root,file), 'r') as zip\_ref:\n",

" #zip\_ref.setpassword(pwd = bytes(pswd, 'utf-8'))\n",

" zip\_ref.extractall()\n",

" os.remove(os.path.join(root,file))\n",

" except:\n",

" os.remove(os.path.join(root,file))\n",

"for root, dirs, files in os.walk('/content/mal'):\n",

" for file in files:\n",

" if file.endswith('.apk'):\n",

" print(file)\n",

" shutil.copy(os.path.join(root,file), '/content/dataset/malign/')\n",

" os.remove(os.path.join(root,file))\n"

]

},

{

"cell\_type": "code",

"execution\_count": 0,

"metadata": {

"colab": {},

"colab\_type": "code",

"id": "28v7MvQjHkgZ"

},

68

"outputs": [],

"source": [

"\n",

"import shutil\n",

"\n",

"msources = ['/content/Adware', '/content/PremiumSMS', '/content/SMS', '/content/Ransomware', '/content/Scareware']\n",

"bsources = ['/content/Benign\_2015', '/content/Benign\_2016', '/content/Benign\_2017']\n",

"for s in msources:\n",

" for root, dirs, files in os.walk(s):\n",

" for file in files:\n",

" if file.endswith('.apk'):\n",

" print(file)\n",

" shutil.copy(os.path.join(root,file), '/content/dataset/malign/')\n",

"for p in bsources:\n",

" for root, dirs, files in os.walk(p):\n",

68

" for file in files:\n",

" if file.endswith('.apk'):\n",

" print(file)\n",

" shutil.copy(os.path.join(root,file), '/content/dataset/benign/')\n",

" os.remove(os.path.join(root,file))"

]

},

{

"cell\_type": "code",

"execution\_count": 0,

"metadata": {

"colab": {},

"colab\_type": "code",

"id": "rU0UX1azTTps"

},

"outputs": [],

"source": [

"!pip install androguard"

]

},

{

"cell\_type": "code",

"execution\_count": 0,

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/",

"height": 34

},

"colab\_type": "code",

"id": "9KN8ubPKgCbW",

"outputId": "a0c1171f-344d-4879-8114-b379238390bb"

},

69

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"1150 529\n"

]

}

],

"source": [

"import os, sys, stat \n",

"malign = '/content/dataset/malign'\n",

"benign = '/content/dataset/benign'\n",

"countM = 0\n",

"countB = 0\n",

"\n",

"for entry in os.scandir(benign):\n",

" if entry.path.endswith(\".apk\") and entry.is\_file():\n",

" countB += 1\n",

" #if countB> 550:\n",

" # os.remove(entry.path)\n",

"for entry in os.scandir(malign):\n",

" if entry.path.endswith(\".apk\") and entry.is\_file():\n",

" countM += 1\n",

" \n",

"print(countB, countM)\n"

]

},

{

"cell\_type": "code",

"execution\_count": 0,

"metadata": {

"colab": {},

"colab\_type": "code",

"id": "bp-JNIdrTcvP"

},

"outputs": [],

"source": [

"import os, sys, stat \n",

"from androguard.core.bytecodes.apk import APK\n",

"\n",

"malign = '/content/dataset/malign'\n",

"benign = '/content/dataset/benign'\n",

"\n",

"permissions = []\n",

"countM = 0\n",

"countB = 0\n",

"\n",

"for entry in os.scandir(benign):\n",

" if entry.path.endswith(\".apk\") and entry.is\_file():\n",

" try:\n",

" a = APK(entry.path)\n",

70

" perm = a.get\_permissions()\n",

" countB += 1\n",

" for per in perm:\n",

" if per not in permissions and per.startswith('android.permission'):\n",

" permissions.append(per)\n",

" except:\n",

" os.remove(entry.path)\n",

"\n",

"for entry in os.scandir(malign):\n",

" if entry.path.endswith(\".apk\") and entry.is\_file():\n",

" try:\n",

" a = APK(entry.path)\n",

" perm = a.get\_permissions()\n",

" countM += 1\n",

" for per in perm:\n",

" if per not in permissions and per.startswith('android.permission'):\n",

" permissions.append(per)\n",

" except:\n",

" os.remove(entry.path) \n"

]

},

{

"cell\_type": "code",

"execution\_count": 0,

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/",

"height": 34

},

"colab\_type": "code",

"id": "1IgtUtEfVUgh",

"outputId": "423e9b1f-0c84-4486-9d3d-3eca2e0d66d1"

},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"1150 529\n"

]

}

],

"source": [

"print(countB,countM)"

]

},

{

"cell\_type": "code",

"execution\_count": 0,

"metadata": {

"colab": {},

"colab\_type": "code",

"id": "jXGT3u-cWUw\_"

71

},

"outputs": [],

"source": [

"import pandas as pd\n",

"default = open('/content/drive/My Drive/Android-Malware-Detection/DefaultPermList.txt','r').readlines()\n",

"perms = [s.rstrip('\\n') for s in default]\n",

"for p in permissions:\n",

" if p not in perms:\n",

" perms.append(p)"

]

},

{

"cell\_type": "code",

"execution\_count": 0,

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/",

"height": 34

},

"colab\_type": "code",

"id": "3AOmlGOqXI2D",

"outputId": "5306ac36-abaf-4dd8-ef8d-00cc6aa5a600"

},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"428\n"

]

}

],

"source": [

"print(len(perms))\n",

"with open('/content/drive/My Drive/Android-Malware-Detection/permissions.txt','w+') as f:\n",

" for p in perms:\n",

" f.write(p + '\\n')"

]

},

{

"cell\_type": "code",

"execution\_count": 0,

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/",

"height": 34

},

"colab\_type": "code",

"id": "BNC\_fUiZrLRu",

"outputId": "99cc8838-20b5-4a12-801f-a8e23db558a4"

},

72

"outputs": [

{

"data": {

"text/plain": [

"428"

]

},

"execution\_count": 60,

"metadata": {

"tags": []

},

"output\_type": "execute\_result"

}

],

"source": [

"pe = []\n",

"with open('/content/drive/My Drive/Android-Malware-Detection/permissions.txt','r') as a:\n",

" content = a.readlines()\n",

" for line in content:\n",

" cur\_perm = line[:-1]\n",

" pe.append(cur\_perm)\n",

"len(pe)\n",

"\n"

]

},

{

"cell\_type": "code",

"execution\_count": 0,

"metadata": {

"colab": {},

"colab\_type": "code",

"id": "KgXs\_YNtXPU9"

},

"outputs": [],

"source": [

"dataset\_df = pd.DataFrame(columns=perms)\n",

"dataset\_b = {}\n",

"id = 0\n",

"for entry in os.scandir(benign):\n",

" if entry.path.endswith(\".apk\") and entry.is\_file():\n",

" a = APK(entry.path)\n",

" perm = a.get\_permissions()\n",

" id += 1\n",

" for d in perms:\n",

" if d in perm:\n",

" dataset\_b[d]=1\n",

" else:\n",

" dataset\_b[d]=0\n",

" dataset\_b['class'] = 'benign'\n",

" dataset\_df=dataset\_df.append(dataset\_b, ignore\_index=True)\n",

"dataset\_m = {}\n",

73

"for entry in os.scandir(malign):\n",

" if entry.path.endswith(\".apk\") and entry.is\_file():\n",

" a = APK(entry.path)\n",

" perm = a.get\_permissions()\n",

" id += 1\n",

" for d in perms:\n",

" if d in perm:\n",

" dataset\_m[d]=1\n",

" else:\n",

" dataset\_m[d]=0\n",

" dataset\_m['class'] = 'malign'\n",

" dataset\_df=dataset\_df.append(dataset\_m, ignore\_index=True)"

]

},

{

"cell\_type": "code",

"execution\_count": 0,

"metadata": {

"colab": {},

"colab\_type": "code",

"id": "Co0XjPOdYhEt"

},

"outputs": [],

"source": [

"dataset\_df.to\_csv('/content/drive/My Drive/Android-Malware-Detection/android\_dataset.csv', index=False)"

]

},

{

"cell\_type": "markdown",

"metadata": {

"colab\_type": "text",

"id": "qKNvlHaQOLvH"

},

"source": [

74

**SYSTEM TESTING & OUTPUT SCREEN**

**5.1 System Testing**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**TYPES OF TESTS**

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

75

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

76

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**5.1.1 Unit Testing:**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

# **5.1.2 Integration Testing**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

77

**5.1.3 Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

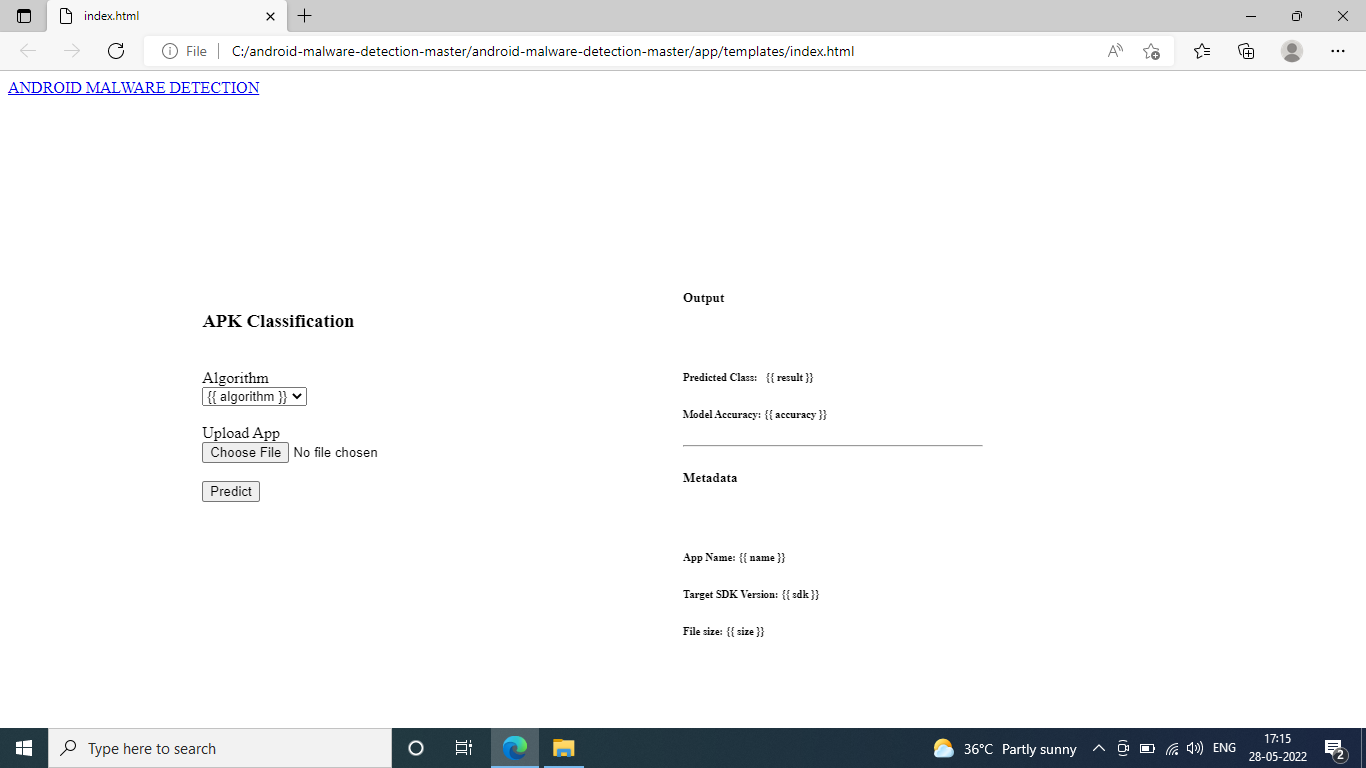
**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**Test Cases**

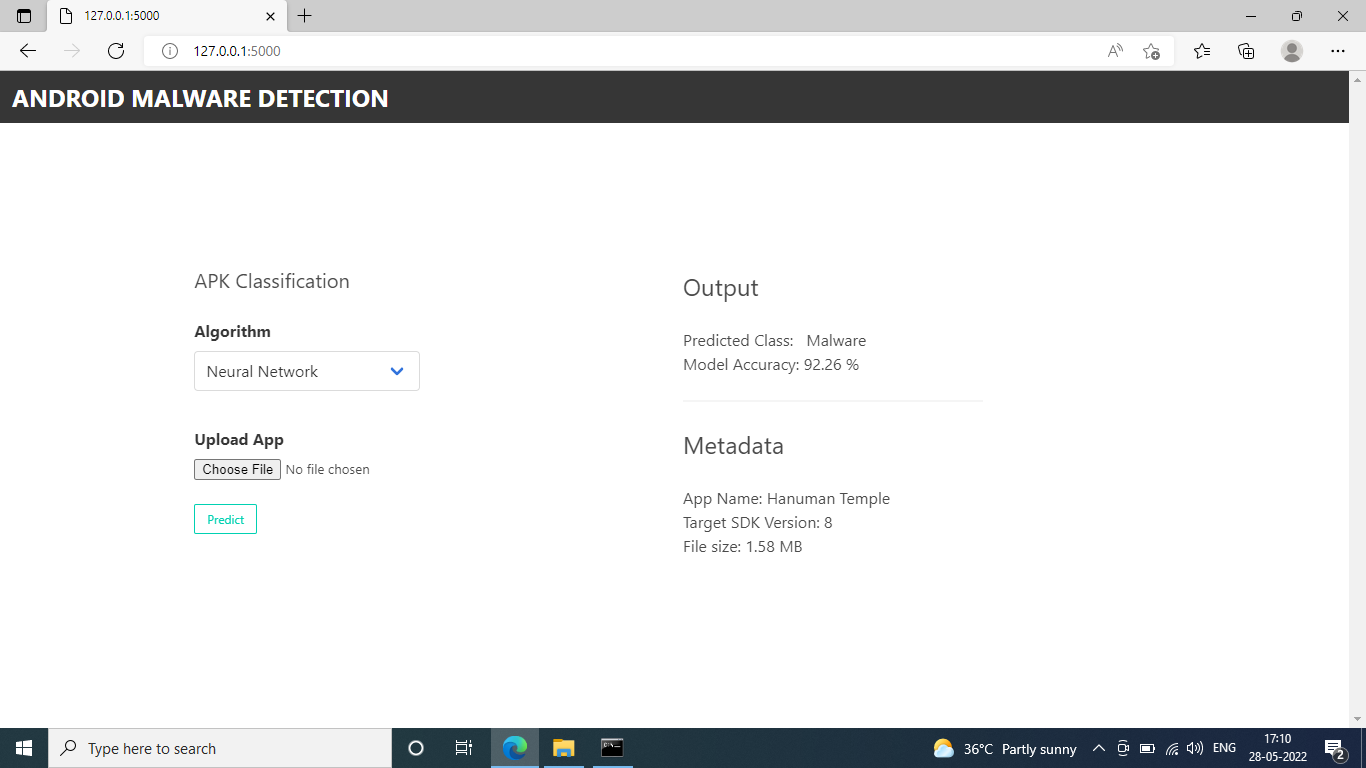
|  |  |  |  |
| --- | --- | --- | --- |
| **TC ID** | **Condition Being Testes** | **Expected Result** | **Result** |
| 1 | Check for null values an duplicate records | Duplicate the duplicate records and null values | passed |
| 2 | Extract features using vectorisation | Display the features of the data | passed |
| 3 | Check if the data is not empty and display records | Show error if it empty else display the data | Passed |
| 4 | Display the number of records for training and test | Display the count of training and test | Passes |
| 5 | Save the model of the best accuracy graph | Saved the current path | passed |

78

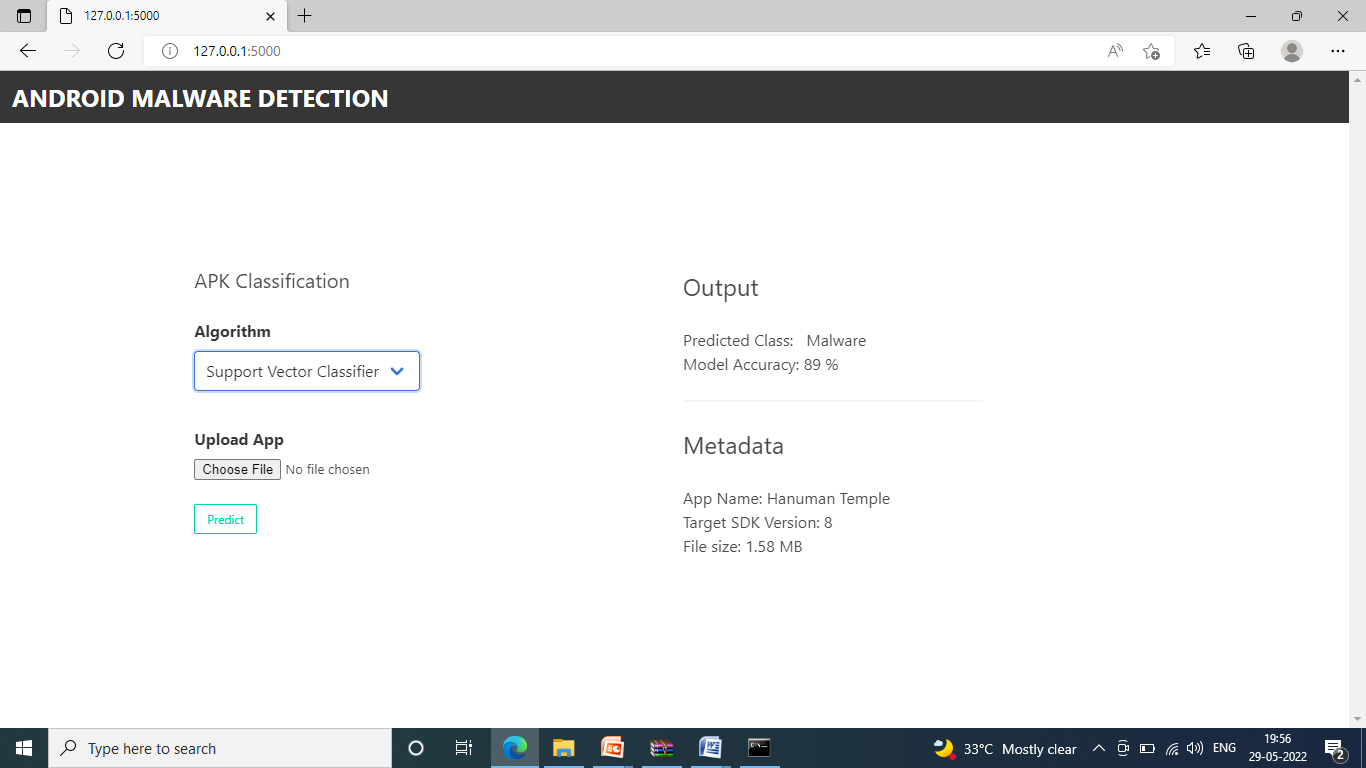
**5.2 Output Screen**



79



80



81

**CONCLUSION & FUTURE WORK**

**6.1 Conclusion**

In our study, we propose category-based machine learning classifiers to improve the performance of the classification models. In static analysis of Android malware, machine learning algorithms have been used to train classifiers with features of malicious apps to build models that capable of detecting malicious patterns. Differently, our classification approach defines legitimate static features for benign apps as opposite to identifying malicious patterns. We utilize the features of the top rated apps in a specific category to define a profile of the common sets of features for that category. In other words, to detect whether or not the app posses the characteristics of benign, we relate between the app’s features and the features that are needed to deliver the category’s functionality that the app belongs to. Android stores organize apps into different categories; 26 categories on the Google Play Store, for example. In each category, the apps deliver a similar functionality as a result the they tend to request a common set of features like same permissions, APIs, hardware components, broadcast receivers, intents filters, etc. On the contrary, malicious apps tend to have abnormal features, less or more than what is common for the category that they belong to. Malicious apps can be identified by comparing between the features they request to the features that are requested by benign apps in the same category. For example, malicious apps, compared to the benign apps in the same category, tend to request over-privileged permissions, listen to specific events that broadcast by the Android system, or using unneeded APIs for the app’s category functionality that can be used to lunch malicious behaviors.

We compare the performance of category-based and non-category based classi- fiers at detecting malicious apps under a specific category. To achieve this comparison, we built three datasets of apps’ features: apps from all categories (allCateg), apps from ”Music & Audio” category (musicCateg), and apps from ”Personalization” category (personaCateg). For each dataset, we trained three machine learning classifiers: Support Vector Machines, Random Forests, and Ada Boost; the classifiers were trained with three group of features: permissions, broadcast receivers, and APIs. For testing, apps from ”Music & Audio” category were tested with (musicCateg) and (allCateg) classifiers, respectively; and apps from ”Personalization” category were tested with personaCateg and allCateg classifiers, as well. The category-based classifiers reported a higher performance compared to the non-category based at detecting malicious and benign in the two categories of our study: ”Music & Audio” and ”Personalization”.

82

**6.2 Future Work**

Our future work will consider three aspects. First, including other static features such as: functions calls in building the classification models to get a better understanding of the processes that apps may lunch in a way to increase the detection accuracy of the classifiers. Second, implementing the proposed solution on a large-scale level by building profile models for other categories and sub categories. Third, testing the feasibility of integrating our solution with dynamic detection techniques by profiling dynamic features for each category; dynamic features like system calls, network connections, resources’ usage, and etc.

83

**BIBLOGRAPHY**

**7.1 Biblography**

1. Androguard usage. <https://code.google.com/p/androguard/wiki/Usage>. Ac- cessed April 24, 2015.
2. Android - statistics & facts — statista. [http://www.statista.com/topics/876/](http://www.statista.com/topics/876/android/) [android/](http://www.statista.com/topics/876/android/). Accessed April 19, 2015.
3. Android and ios continue to dominate the worldwide smartphone market with an- droid shipments just shy of 800 million in 2013, according to idc. [http://www.idc.](http://www.idc.com/getdoc.jsp?containerId=prUS24676414) [com/getdoc.jsp?containerId=prUS24676414](http://www.idc.com/getdoc.jsp?containerId=prUS24676414). Accessed April 19, 2015.
4. Application fundamentals — android developers. [http://developer.android.com/](http://developer.android.com/guide/components/fundamentals.html) [guide/components/fundamentals.html](http://developer.android.com/guide/components/fundamentals.html). Accessed April 19, 2015.
5. Are — download/installation. [https://redmine.honeynet.org/projects/are/](https://redmine.honeynet.org/projects/are/wiki) [wiki](https://redmine.honeynet.org/projects/are/wiki). Accessed April 28, 2015.
6. Dynamic analysis tools for android fail to detect malware with heuristic evasion techniques. [http://thehackernews.com/2014/05/](http://thehackernews.com/2014/05/dynamic-analysis-tools-for-android-fail.html) [dynamic-analysis-tools-for-android-fail.html](http://thehackernews.com/2014/05/dynamic-analysis-tools-for-android-fail.html). Accessed April 19, 2015.
7. ”global smartphone sales exceed 1.2b units in 2014.” gfk - we see the big picture. [http://www.gfk.com/news-and-events/press-room/press-releases/](http://www.gfk.com/news-and-events/press-room/press-releases/pages/global-smartphone-sales-exceed-1-2b-units-in-2014.aspx) [pages/global-smartphone-sales-exceed-1-2b-units-in-2014.aspx.](http://www.gfk.com/news-and-events/press-room/press-releases/pages/global-smartphone-sales-exceed-1-2b-units-in-2014.aspx) Accessed April 19, 2015.
8. Google :We have 1 billion monthly active android users[http://www.businessinsider.com/](http://www.businessinsider.com/google-we-have-1-billion-monthly-active-android-users-2014-6) [google-we-have-1-billion-monthly-active-android-users-2014-6.](http://www.businessinsider.com/google-we-have-1-billion-monthly-active-android-users-2014-6) Accessed April 19, 2015.
9. Report: 97- forbes. [http://www.forbes.com/sites/gordonkelly/2014/03/24/](http://www.forbes.com/sites/gordonkelly/2014/03/24/report-97-of-mobile-malware-is-on-android-this-is-the-easy-way-you-stay-safe/)

[report-97-of-mobile-malware-is-on-android-this-is-the-easy-way-you-](http://www.forbes.com/sites/gordonkelly/2014/03/24/report-97-of-mobile-malware-is-on-android-this-is-the-easy-way-you-stay-safe/)stay-safe/

[..](http://www.forbes.com/sites/gordonkelly/2014/03/24/report-97-of-mobile-malware-is-on-android-this-is-the-easy-way-you-stay-safe/) Accessed April 19, 2015.

84

1. Smartphone os market share, q4 2014. [http://www.idc.com/prodserv/](http://www.idc.com/prodserv/smartphone-os-market-share.jsp) [smartphone-os-market-share.jsp](http://www.idc.com/prodserv/smartphone-os-market-share.jsp). Accessed April 19, 2015.
2. Arp, D., Spreitzenbarth, M., Hu¨bner, M., Gascon, H., Rieck, K., and Siemens,

C. (2014). Drebin: Effective and explainable detection of android malware in your pocket. In *Proc. of NDSS*.

1. Aung, Z. and Zaw, W. (2013). Permission-based android malware detection. *Inter- national Journal of Scientific and Technology Research*, 2(3):228–234.
2. Burguera, I., Zurutuza, U., and Nadjm-Tehrani, S. (2011). Crowdroid: behavior- based malware detection system for android. In *Proceedings of the 1st ACM workshop on Security and privacy in smart phones and mobile devices*, pages 15–26. ACM.
3. Enck, W., Gilbert, P., Han, S., Tendulkar, V., Chun, B.-G., Cox, L. P., Jung, J., McDaniel, P., and Sheth, A. N. (2014). Taintdroid: an information-flow tracking sys- tem for real time privacy monitoring on smart phones. *ACM Transactions on Computer Systems (TOCS)*, 32(2):5.
4. Enck, W., Ongtang, M., and McDaniel, P. (2009). On lightweight mobile phone application certification. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 235–245. ACM.
5. Felt, A. P., Greenwood, K., and Wagner, D. (2011). The effectiveness of application permissions. In *Proceedings of the 2nd USENIX conference on Web application development*, pages 7–7.
6. Felt, A. P., Ha, E., Egelman, S., Haney, A., Chin, E., and Wagner, D. (2012). Android permissions: User attention, comprehension, and behavior. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, page 3. ACM.
7. Fuchs, A. P., Chaudhuri, A., and Foster, J. S. (2009). Scandroid: Automated security certification of android applications. *Manuscript, Univ. of Maryland,* [*http://www.*](http://www/) *cs. umd. edu/avik/projects/scandroidascaa*, 2(3).
8. Grace, M. C., Zhou, W., Jiang, X., and Sadeghi, A.-R. (2012). Unsafe exposure analysis of mobile in-app advertisements. In *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks*, pages 101–112. ACM.
9. Hein, C. L. P. M. (2014). Static based malware detection framework for android application. *signature*, 1(1).
10. Liu, L., Yan, G., Zhang, X., and Chen, S. (2009). Virusmeter: Preventing your cell phone from spies. In *Recent Advances in Intrusion Detection*, pages 244–264.

86