**Android Malware Detection Using Genetic Algorithm based Optimized Feature Selection and Machine Learning**

**1. INTRODUCTION:**

Android Apps are freely available on Google Playstore, the official Android app store as well as third-party app stores for users to download. Due to its open source nature and popularity, malware writers are increasingly focusing on developing malicious applications for Android operating system. In spite of various attempts by Google Playstore to protect against malicious apps, they still find their way to mass market and cause harm to users by misusing personal information related to their phone book, mail accounts, GPS location information and others for misuse by third parties or else take control of the phones remotely. Therefore, there is need to perform malware analysis or reverse-engineering of such malicious applications which pose serious threat to Android platforms. Broadly speaking, Android Malware analysis is of two types: Static Analysis and Dynamic Analysis. Static analysis basically involves analyzing the code structure without executing it while dynamic analysis is examination of the runtime behavior of Android Apps in constrained environment. Given in to the ever-increasing variants of Android Malware posing zero-day threats, an efficient mechanism for detection of Android malwares is required. In contrast to signature-based approach which requires regular update of signature database, machine-learning based approach in combination with static and dynamic analysis can be used to detect new variants of Android Malware posing zero-day threats. In, broad yet lightweight static analysis has been performed achieving a decent detection accuracy of 94% using Support Vector Machine algorithm. Nikola Milosevic et al. presented static analysis based classification through two methodologies: one was permissions based while the other involved representation of the source code as a bag of words. Another approach based on identifying most significant permissions and applying machine learning on it for evaluation has been proposed. MADAM provides an multi-level analysis framework where behavior of Android Apps is captured upto four levels: from package, user, application to kernel level, achieving detection accuracy as high as 96% with one disadvantage being that it could run only on rooted devices, making it incapable for commercial use. SAMADroid proposed a three-way novel host server based methodology for improved performance as far as asset usage is concerned for malware detection on mobile devices. The current drift in malware detection has shifted towards deep learning applications where it requires least human intervention as proposed. An important step in all machine learning based approaches is feature selection. Obtaining optimal feature set will not only help in improving experimentation results but will also help in reducing the curse of dimensionality associated with most machine learning based algorithms. Fest proposed a novel and efficient algorithm for feature selection to improve overall detection accuracy. In, a review of various feature selection algorithms for malware detection has been presented providing guidelines for selection. In the proposed work, Genetic algorithm has been used because of its capabilities in finding a feature subset selected from original feature vector such that it gives the best accuracy for classifiers on which they are trained. It has been used, previously also, in combination with machine learning and deep learning algorithms to obtain the most optimal feature subset. The main contribution of the work is reduction of feature dimension to less than half of original feature-set using Genetic Algorithm such that it can be fed as input to machine learning classifiers for training with reduced complexity while maintaining their accuracy in malware classification. In contrast to exhaustive method of feature selection which requires testing for 2N different combinations, where N is the number of features, Genetic Algorithm, a heuristic searching approach based on fitness function has been used for feature selection. The optimized feature set obtained using Genetic algorithm is used to train two machine learning algorithms: Support Vector Machine and Neural Network. It is observed that a decent classification accuracy of more than 94% is maintained while working on a much lower feature dimension, thereby, reducing the training time complexity of classifiers. The remaining paper is structured as follows: Section II discusses about the proposed methodology used. Section III presents experimentation results obtained by applying genetic algorithm for feature selection to train machine learning algorithms. Section IV briefs about the general conclusions drawn from the experimentations.

**1.1 Objective of the project:**

Android platform due to open source characteristic and Google backing has the largest global market share. Being the world’s most popular operating system, it has drawn the attention of cyber criminals operating particularly through wide distribution of malicious applications. This paper proposes an effectual machine-learning based approach for Android Malware Detection making use of evolutionary Genetic algorithm for discriminatory feature selection. Selected features from Genetic algorithm are used to train machine learning classifiers and their capability in identification of Malware before and after feature selection is compared. The experimentation results validate that Genetic algorithm gives most optimized feature subset helping in reduction of feature dimension to less than half of the original feature-set. Classification accuracy of more than 94% is maintained post feature selection for the machine learning based classifiers, while working on much reduced feature dimension, thereby, having a positive impact on computational complexity of learning classifiers.

**2. LITERATURE SURVEY:**

**“Drebin: Effective and Explainable Detection of Android Malware in Your Pocket,”**

Malicious applications pose a threat to the security of the Android platform. The growing amount and diversity of these applications render conventional defenses largely ineffective and thus Android smartphones often remain un-protected from novel malware. In this paper, we propose DREBIN, a lightweight method for detection of Android malware that enables identifying malicious applications di-rectly on the smartphone. As the limited resources impede monitoring applications at run-time, DREBIN performs a broad static analysis, gathering as many features of an ap-plication as possible. These features are embedded in a joint vector space, such that typical patterns indicative for malware can be automatically identified and used for ex-plaining the decisions of our method. In an evaluation with 123,453 applications and 5,560 malware samples DREBIN outperforms several related approaches and detects 94% of the malware with few false alarms, where the explana-tions provided for each detection reveal relevant properties of the detected malware. On five popular smartphones, the method requires 10 seconds for an analysis on average, ren-dering it suitable for checking downloaded applications di-rectly on the device.

**“Machine learning aided Android malware classification,”**

The widespread adoption of Android devices and their capability to access significant private and confidential information have resulted in these devices being targeted by malware developers. Existing Android malware analysis techniques can be broadly categorized into static and dynamic analysis. In this paper, we present two machine learning aided approaches for static analysis of Android malware. The first approach is based on permissions and the other is based on source code analysis utilizing a bag-of-words representation model. Our permission-based model is computationally inexpensive, and is implemented as the feature of OWASP Seraphim droid Android app that can be obtained from Google Play Store. Our evaluations of both approaches indicate an F-score of 95.1% and F-measure of 89% for the source code-based classification and permission-based classification models, respectively.

**“Significant Permission Identification for Machine-Learning-Based Android Malware Detection,”**

The alarming growth rate of malicious apps has become a serious issue that sets back the prosperous mobile ecosystem. A recent report indicates that a new malicious app for Android is introduced every 10 s. To combat this serious malware campaign, we need a scalable malware detection approach that can effectively and efficiently identify malware apps. Numerous malware detection tools have been developed, including system-level and network-level approaches. However, scaling the detection for a large bundle of apps remains a challenging task. In this paper, we introduce Significant Permission IDentification (SigPID), a malware detection system based on permission usage analysis to cope with the rapid increase in the number of Android malware. Instead of extracting and analyzing all Android permissions, we develop three levels of pruning by mining the permission data to identify the most significant permissions that can be effective in distinguishing between benign and malicious apps. SigPID then utilizes machine-learning-based classification methods to classify different families of malware and benign apps. Our evaluation finds that only 22 permissions are significant. We then compare the performance of our approach, using only 22 permissions, against a baseline approach that analyzes all permissions. The results indicate that when a support vector machine is used as the classifier, we can achieve over 90% of precision, recall, accuracy, and F-measure, which are about the same as those produced by the baseline approach while incurring the analysis times that are 4-32 times less than those of using all permissions. Compared against other state-of-the-art approaches, SigPID is more effective by detecting 93.62% of malware in the dataset and 91.4% unknown/new malware samples.

**“MADAM: Effective and Efficient Behavior-based Android Malware Detection and Prevention,”**

Android users are constantly threatened by an increasing number of malicious applications (apps), generically called malware. Malware constitutes a serious threat to user privacy, money, device and file integrity. In this paper we note that, by studying their actions, we can classify malware into a small number of behavioral classes, each of which performs a limited set of misbehaviors that characterize them. These misbehaviors can be defined by monitoring features belonging to different Android levels. In this paper we present MADAM, a novel host-based malware detection system for Android devices which simultaneously analyzes and correlates features at four levels: kernel, application, user and package, to detect and stop malicious behaviors. MADAM has been specifically designed to take into account those behaviors that are characteristics of almost every real malware which can be found in the wild. MADAM detects and effectively blocks more than 96 percent of malicious apps, which come from three large datasets with about 2,800 apps, by exploiting the cooperation of two parallel classifiers and a behavioral signature-based detector. Extensive experiments, which also include the analysis of a testbed of 9,804 genuine apps, have been conducted to show the low false alarm rate, the negligible performance overhead and limited battery consumption.

**“SAMADroid: A Novel 3-Level Hybrid Malware Detection Model for Android Operating System,”**

For the last few years, Android is known to be the most widely used operating system and this rapidly increasing popularity has attracted the malware developer's attention. Android allows downloading and installation of apps from other unofficial market places. This gives malware developers an opportunity to put repackaged malicious applications in third-party app-stores and attack the Android devices. A large number of malware analysis and detection systems have been developed which uses static analysis, dynamic analysis, or hybrid analysis to keep Android devices secure from malware. However, the existing research clearly lags in detecting malware efficiently and accurately. For accurate malware detection, multilayer analysis is required which consumes large amount of hardware resources of resource constrained mobile devices. This research proposes an efficient and accurate solution to this problem, named SAMADroid, which is a novel 3-level hybrid malware detection model for Android operating systems. The research contribution includes multiple folds. First, many of the existing Android malware detection techniques are thoroughly investigated and categorized on the basis of their detection methods. Also, their benefits along with limitations are deduced. A novel 3-level hybrid malware detection model for Android operating systems is developed, that can provide high detection accuracy by combining the benefits of the three different levels: 1) Static and Dynamic Analysis; 2) Local and Remote Host; and 3) Machine Learning Intelligence. Experimental results show that SAMADroid achieves high malware detection accuracy by ensuring the efficiency in terms of power and storage consumption.

**“A Multimodal Deep Learning Method for Android Malware Detection using Various Features,”**

With the widespread use of smartphones, the number of malware has been increasing exponentially. Among smart devices, android devices are the most targeted devices by malware because of their high popularity. This paper proposes a novel framework for android malware detection. Our framework uses various kinds of features to reflect the properties of android applications from various aspects, and the features are refined using our existence-based or similarity-based feature extraction method for effective feature representation on malware detection. Besides, a multimodal deep learning method is proposed to be used as a malware detection model. This paper is the first study of the multimodal deep learning to be used in the android malware detection. With our detection model, it was possible to maximize the benefits of encompassing multiple feature types. To evaluate the performance, we carried out various experiments with a total of 41 260 samples. We compared the accuracy of our model with that of other deep neural network models. Furthermore, we evaluated our framework in various aspects including the efficiency in model updates, the usefulness of diverse features, and our feature representation method. In addition, we compared the performance of our framework with those of other existing methods including deep learning-based methods.

**“Evolving Deep Neural Networks architectures for Android malware classification,”**

Deep Neural Networks (DNN) have become a powerful, widely used, and successful mechanism to solve problems of different nature and varied complexity. Their ability to build models adapted to complex non-linear problems, have made them a technique widely applied and studied. One of the fields where this technique is currently being applied is in the malware classification problem. The malware classification problem has an increasing complexity, due to the growing number of features needed to represent the behaviour of the application as exhaustively as possible. Although other classification methods, as those based on SVM, have been traditionally used, the DNN pose a promising tool in this field. However, the parameters and architecture setting of these DNNs present a serious restriction, due to the necessary time to find the most appropriate configuration. This paper proposes a new genetic algorithm designed to evolve the parameters, and the architecture, of a DNN with the goal of maximising the malware classification accuracy, and minimizing the complexity of the model. This model is tested against a dataset of malware samples, which are represented using a set of static features, so the DNN has been trained to perform a static malware classification task. The experiments carried out using this dataset show that the genetic algorithm is able to select the parameters and the DNN architecture settings, achieving a 91% accuracy.

**“A Deep Learning Approach to Android Malware Feature Learning and Detection,”**

The growing amount and diversity of Android malware has significantly weakened the effectiveness of the conventional defense mechanisms, and thus Android platform often remains unprotected from new and unknown malware. To address these limitations, we propose DroidDeep, a malware detection approach for the Android platform based on the deep learning model. Deep learning emerges as a new area of machine learning research that has attracted increasing attention in artificial intelligence. To implement this, we first extract five types of features from the static analysis of Android apps. Then, we build the deep learning model to learn features from Android apps. Finally, the learned features are used to detect unknown Android malware. In an experiment with 3,986 benign apps and 3,986 malware, DroidDeep outperforms several existing malware detection approaches and achieves a 99.4% detection accuracy. Moreover, DroidDeep can achieve a remarkable run-time efficiency which makes it very easy to adapt to a lager scale of real-world Android malware detection.

**“Fest : A Feature Extraction and Selection Tool for Android Malware Detection,”**

Android has become one of the most popular mobile operating systems because of numerous applications (apps) it provides. However, Android malware downloaded from third-party markets threatens users' privacy, and most of them remain undetected because of the lack of efficient and accurate detecting techniques. Prior efforts on Android malware detection attempted to build precise classification models by manually choosing features, and few of them has used any feature selection algorithms to help pick typical features. In this paper, we present Feature Extraction and Selection Tool (Fest), a feature-based machine learning approach for malware detection. We first implement a feature extraction tool, AppExtractor, which is designed to extract features, such as permissions or APIs, according to the predefined rules. Then we propose a feature selection algorithm, FrequenSel. Unlike existing selection algorithms which pick features by calculating their importance, FrequenSel selects features by finding the difference their frequencies between malware and benign apps, because features which are frequently used in malware and rarely used in benign apps are more important to distinguish malware from benign apps. In experiments, we evaluate our approach with 7972 apps, and the results show that Fest gets nearly 98% accuracy and recall, with only 2% false alarms. Moreover, Fest only takes 6.5s to analyze an app on a common PC, which is very time-efficient for malware detection in Android markets.

**“A review on feature selection in mobile malware detection,”**

The widespread use of mobile devices in comparison to personal computers has led to a new era of information exchange. The purchase trends of personal computers have started decreasing whereas the shipment of mobile devices is increasing. In addition, the increasing power of mobile devices along with portability characteristics has attracted the attention of users. Not only are such devices popular among users, but they are favorite targets of attackers. The number of [mobile malware](https://www.sciencedirect.com/topics/computer-science/mobile-malware) is rapidly on the rise with [malicious activities](https://www.sciencedirect.com/topics/computer-science/malicious-activity), such as stealing users data, sending premium messages and making phone call to premium numbers that users have no knowledge. Numerous studies have developed methods to thwart such attacks. In order to develop an effective detection system, we have to select a subset of features from hundreds of available features. In this paper, we studied 100 research works published between 2010 and 2014 with the perspective of feature selection in mobile [malware detection](https://www.sciencedirect.com/topics/computer-science/malware-detection). We categorize available features into four groups, namely, static features, dynamic features, hybrid features and [applications metadata](https://www.sciencedirect.com/topics/computer-science/application-metadata). Additionally, we discuss datasets used in the recent research studies as well as analyzing evaluation measures utilized.

**3. SYSTEM ANALYSIS**

**3.1 Existing System**

The main contribution of the work is reduction of feature dimension to less than half of original feature-set using Genetic Algorithm such that it can be fed as input to machine learning classifiers for training with reduced complexity while maintaining their accuracy in malware classification. In contrast to exhaustive method of feature selection which requires testing for 2N different combinations, where N is the number of features, Genetic Algorithm, a heuristic searching approach based on fitness function has been used for feature selection. The optimized feature set obtained using Genetic algorithm is used to train two machine learning algorithms: Support Vector Machine and Neural Network. It is observed that a decent classification accuracy of more than 94% is maintained while working on a much lower feature dimension, thereby, reducing the training time complexity of classifiers.

apple’s automatic visual classification problem.

**Disadvantages of Existing System:**

1. Less Accuracy.

**3.2Proposed System**

Two set of Android Apps or APKs: Malware/Goodware is reverse engineered to extract features such as permissions and count of App Components such as Activity, Services, Content Providers, etc. These features are used as featurevector with class labels as Malware and Goodware represented by 0 and 1 respectively in CSV format. To reduce dimensionality of feature-set, the CSV is fed to Genetic Algorithm to select the most optimized set of features. The optimized set of features obtained is used for training two machine learning classifiers: Support Vector Machine and Neural Network. In the proposed methodology, static features are obtained from AndroidManifest.xml which contains all the important information needed by any Android platform about the Apps. Androguard tool has been used for disassembling of the APKs and getting the static features.

**Advantages of Proposed System:**

1. Accuracy is more

**Modules:**

1. Upload Android dataset
2. Generate Train & test model
3. Run SVM & Neural network algorithms
4. Run SVM & Neural network algorithms with Genetic Algorithm
5. Display Accuracy Graph
6. Execution Time Graph

**Module description:**

**Upload Android dataset:**

Upload Android Malware Dataset module is used to upload Android dataset.

**Generate Train & test model:**

Generate Train & Test Model module is used to split dataset into train and test part. All machine learning algorithms will take 80% dataset for training and 20% dataset to test accuracy of trained model. After clicking that button will get train and test model. We can see there are total 3799 android app records are there and application using 3039 records for training and 760 records for testing. Now we have both train and test model.

**Run SVM & Neural network algorithms**

Run SVM Algorithm to generate SVM model on train and test and get its accuracy, we got 98% accuracy for SVM. Run Neural Network Algorithm to test neural network accuracy. Neural network also gave 98.64% accuracy.

**Run SVM & Neural network algorithms with Genetic Algorithm**

Run SVM with Genetic Algorithm to choose optimize features and then run SVM on optimize features to get accuracy SVM with Genetic algorithm got 93% accuracy. Genetic with SVM accuracy is less but its execution time will be less which we can see at the time of comparison graph. Now run Neural Network with Genetic Algorithm to get NN accuracy with genetic algorithm. NN with genetic got 98.02% accuracy.

**Display Accuracy Graph**

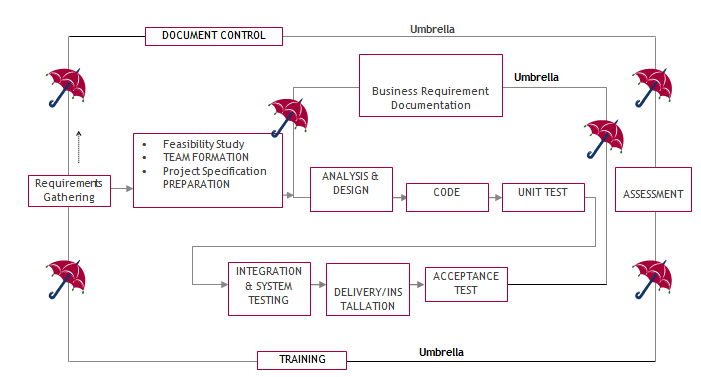
Accuracy graph module to see all algorithms accuracy in graph. The graph x-axis represents algorithm name and y-axis represents accuracy and in all SVM got high accuracy.

**Execution Time Graph**

Execution Time Graph module to get execution time of all algorithms. In above graph x-axis represents algorithm name and y-axis represents execution time. From above graph we can conclude that with genetic algorithm machine learning algorithms taking less time to build model.

**3.3. PROCESS MODEL USED WITH JUSTIFICATION**

**SDLC (Umbrella Model):**



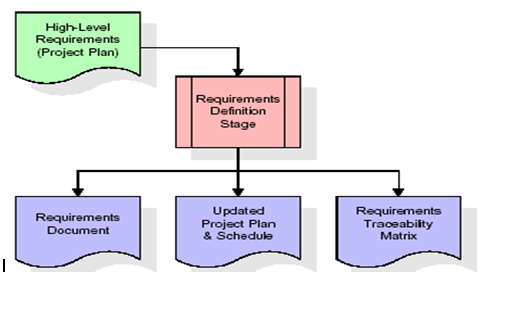
SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

**Stages in SDLC:**

* Requirement Gathering
* Analysis
* Designing
* Coding
* Testing
* Maintenance

**Requirements Gathering** **stage:**

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.



These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are notincluded in the requirements document.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.

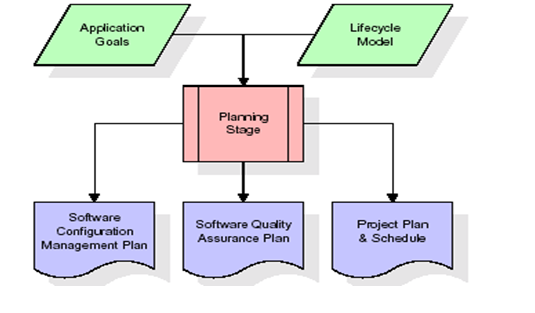
In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term requirements traceability.

The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

* Feasibility study is all about identification of problems in a project.
* No. of staff required to handle a project is represented as Team Formation, in this case only modules are individual tasks will be assigned to employees who are working for that project.
* Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator.

**Analysis Stage:**

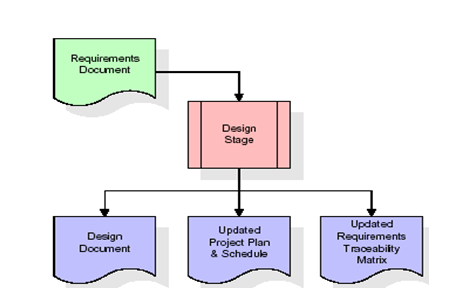
The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.



The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high level estimates of effort for the out stages.

**Designing Stage:**

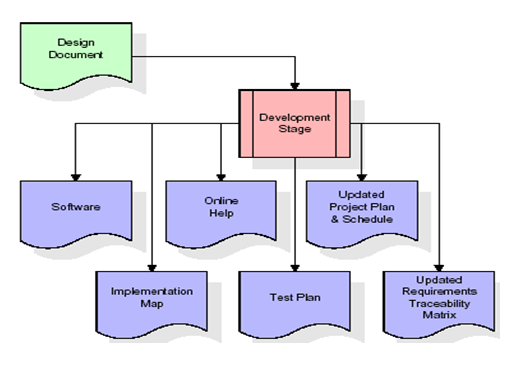
The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.



When the design document is finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement. The outputs of the design stage are the design document, an updated RTM, and an updated project plan.

**Development (Coding) Stage:**

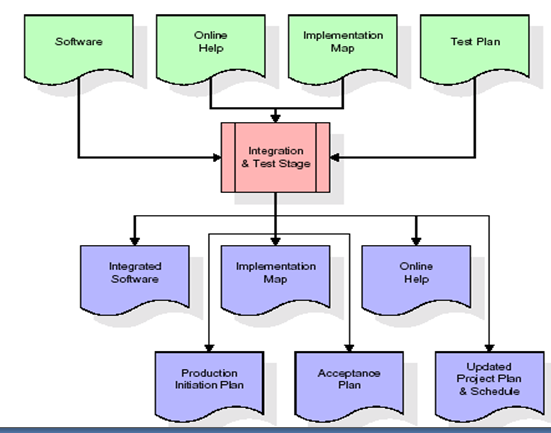
The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artifacts will be produced. Software artifacts include but are not limited to menus, dialogs, and data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.



The RTM will be updated to show that each developed artifact is linked to a specific design element, and that each developed artifact has one or more corresponding test case items. At this point, the RTM is in its final configuration. The outputs of the development stage include a fully functional set of software that satisfies the requirements and design elements previously documented, an online help system that describes the operation of the software, an implementation map that identifies the primary code entry points for all major system functions, a test plan that describes the test cases to be used to validate the correctness and completeness of the software, an updated RTM, and an updated project plan.

**Integration & Test Stage:**

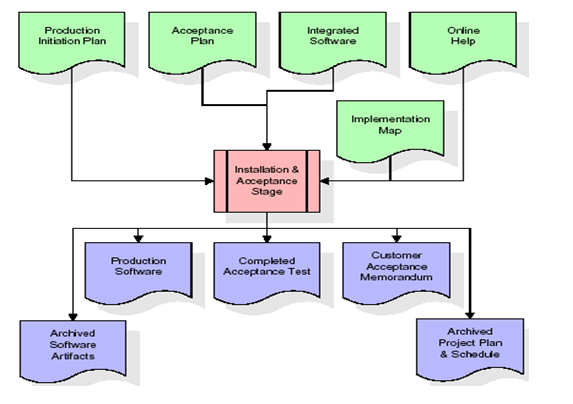
During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.



The outputs of the integration and test stage include an integrated set of software, an online help system, an implementation map, a production initiation plan that describes reference data and production users, an acceptance plan which contains the final suite of test cases, and an updated project plan.

* **Installation & Acceptance Test:**

During the installation and acceptance stage, the software artifacts, online help, and initial production data are loaded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer. After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery of the software.



The primary outputs of the installation and acceptance stage include a production application, a completed acceptance test suite, and a memorandum of customer acceptance of the software. Finally, the PDR enters the last of the actual labor data into the project schedule and locks the project as a permanent project record. At this point the PDR "locks" the project by archiving all software items, the implementation map, the source code, and the documentation for future reference.

**Maintenance:**

Outer rectangle represents maintenance of a project, Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will undergo training on that particular assigned category. For this life cycle there is no end, it will be continued so on like an umbrella (no ending point to umbrella sticks).

**3.4. Software Requirement Specification**

**3.4.1. Overall Description**

A Software Requirements Specification (SRS) – a [requirements specification](http://en.wikipedia.org/wiki/Requirements_specification) for a [software system](http://en.wikipedia.org/wiki/Software_system) is a complete description of the behavior of a system to be developed. It includes a set of [use cases](http://en.wikipedia.org/wiki/Use_case) that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. [Nonfunctional requirements](http://en.wikipedia.org/wiki/Non-functional_requirements) are requirements which impose constraints on the design or implementation (such as [performance engineering](http://en.wikipedia.org/wiki/Performance_engineering) requirements, [quality](http://en.wikipedia.org/wiki/Quality_%28business%29) standards, or design constraints).

System requirements specification: A structured collection of information that embodies the requirements of a system. A [business analyst](http://en.wikipedia.org/wiki/Business_analyst), sometimes titled [system analyst](http://en.wikipedia.org/wiki/System_analyst), is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the [systems development lifecycle](http://en.wikipedia.org/wiki/Systems_development_life_cycle) domain, the BA typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers. Projects are subject to three sorts of requirements:

* [Business requirements](http://en.wikipedia.org/wiki/Business_requirements) describe in business terms *what* must be delivered or accomplished to provide value.
* Product requirements describe properties of a system or product (which could be one of several ways to accomplish a set of business requirements.)
* Process requirements describe activities performed by the developing organization. For instance, process requirements could specify .Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

**ECONOMIC FEASIBILITY**

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

**Operational Feasibility**

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization’s operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

**TECHNICAL FEASIBILITY**

Earlier no system existed to cater to the needs of ‘Secure Infrastructure Implementation System’. The current system developed is technically feasible. It is a web based user interface for audit workflow at NIC-CSD. Thus it provides an easy access to .the users. The database’s purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security.

**3.4.2. External Interface Requirements**

**User Interface**

The user interface of this system is a user friendly python Graphical User Interface.

**Hardware Interfaces**

The interaction between the user and the console is achieved through python capabilities.

**Software Interfaces**

The required software is python.

**Operating Environment**

Windows XP.

**HARDWARE REQUIREMENTS:**

# Processor - Pentium –IV

* Speed - 1.1 Ghz
* RAM - 256 MB(min)
* Hard Disk - 20 GB
* Key Board - Standard Windows Keyboard
* Mouse - Two or Three Button Mouse
* Monitor - SVGA

**SOFTWARE REQUIREMENTS:**

* Operating System - Windows7/8
* Programming Language - Python

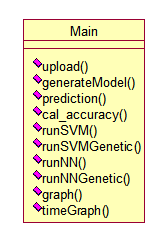
**4. SYSTEM DESIGN:**

**UML Diagram:**

**Class Diagram:**

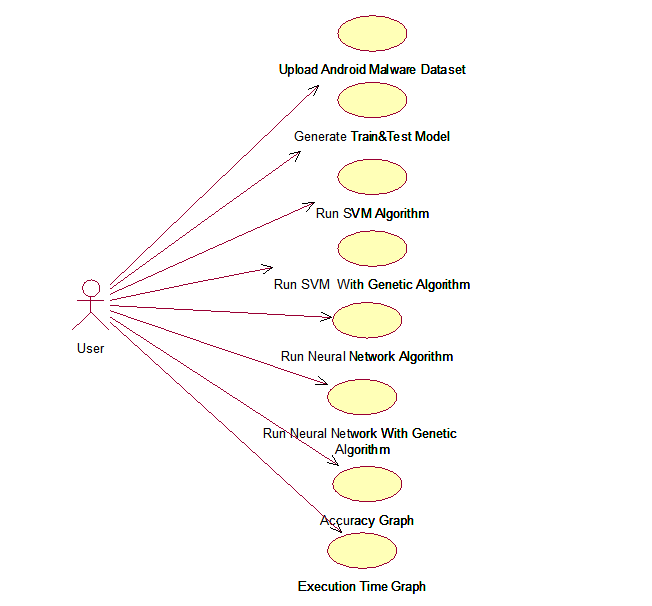
The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. In the diagram, classes are represented with boxes which contain three parts:

* The upper part holds the name of the class
* The middle part contains the attributes of the class
* The bottom part gives the methods or operations the class can take or undertake



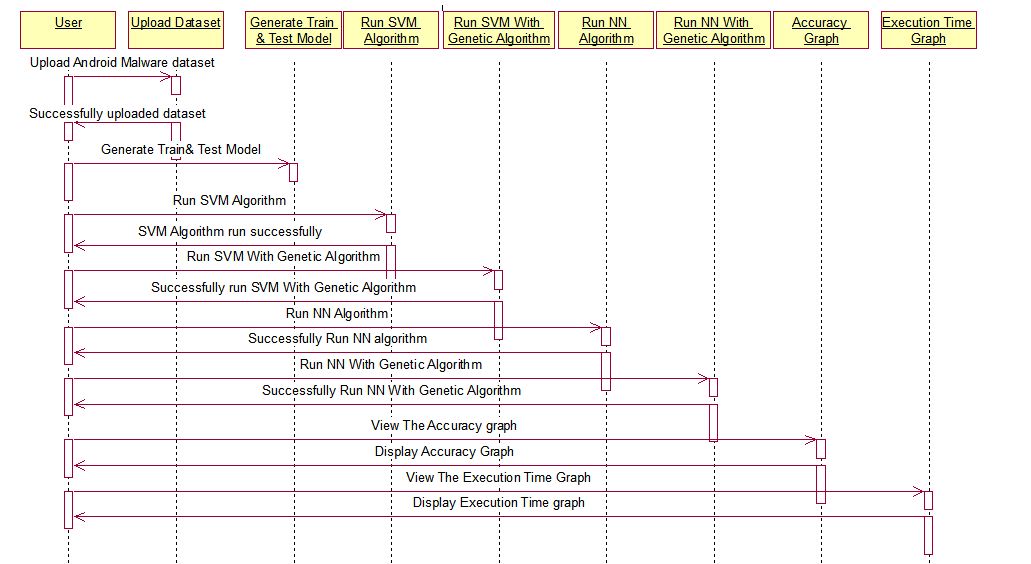
**Use case Diagram:**

A use case diagram at its simplest is a representation of a user's interaction can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.



**Sequence diagram:**

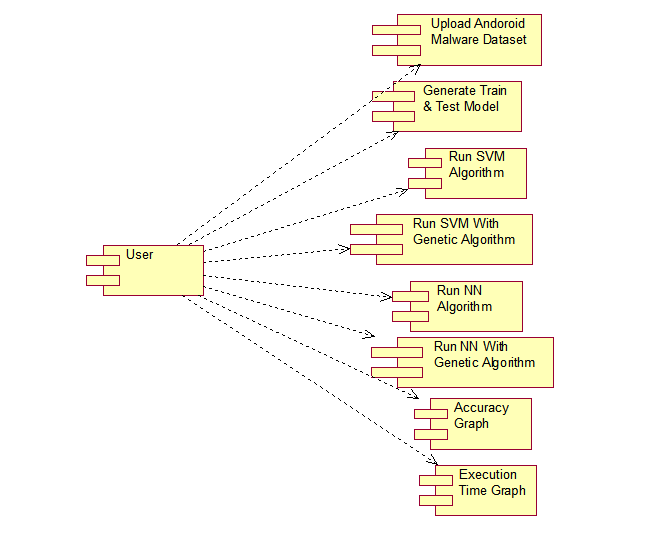
A sequencediagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



**Component Diagram:**

In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems.

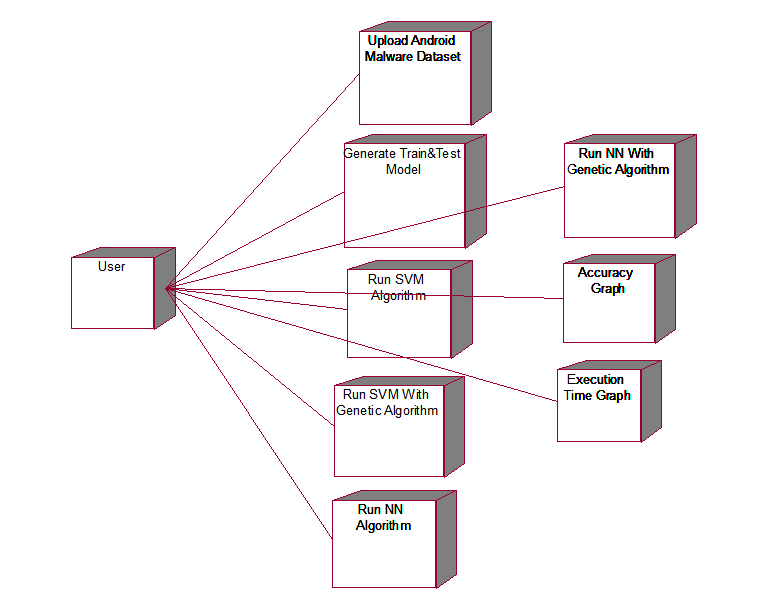
Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components.



**Deployment Diagram:**

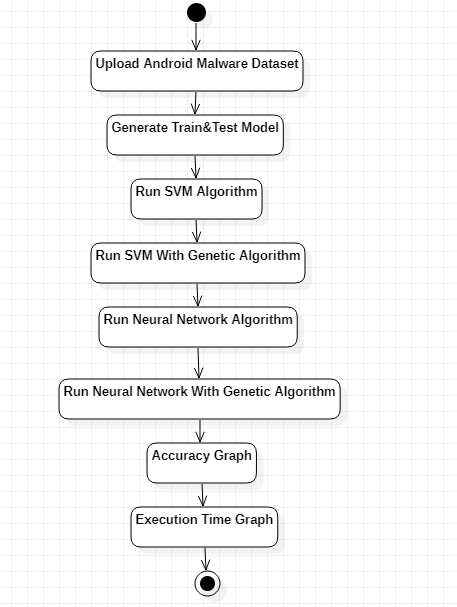
A deployment diagram in the Unified Modeling Language models the *physical* deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI).

The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.



**Activity Diagram:**

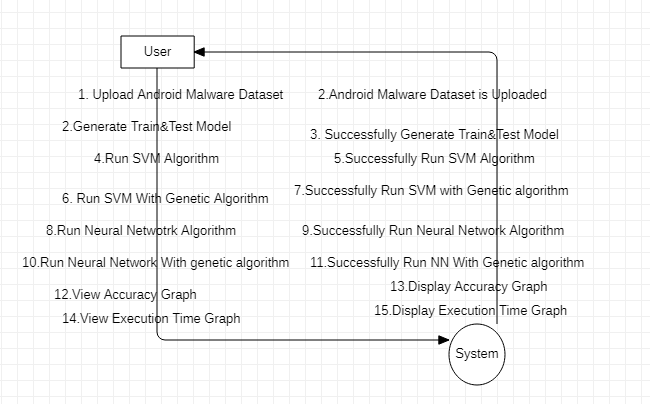
Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent.



**Data Flow Diagram:**

Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs. Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. This analysis can be carried out in precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.

As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD is a model for constructing and analyzing information processes. DFD illustrates the flow of information in a process depending upon the inputs and outputs. A DFD can also be referred to as a Process Model. A DFD demonstrates business or technical process with the support of the outside data saved, plus the data flowing from the process to another and the end results.



**5. IMPLEMETATION:**

**5.1 Python**

Python is a general-purpose language. It has wide range of applications from Web development (like: Django and Bottle), scientific and mathematical computing (Orange, SymPy, NumPy) to desktop graphical user Interfaces (Pygame, Panda3D). The syntax of the language is clean and length of the code is relatively short. It's fun to work in Python because it allows you to think about the problem rather than focusing on the syntax.

**History of Python:**

Python is a fairly old language created by Guido Van Rossum. The design began in the late 1980s and was first released in February 1991.

**Why Python was created?**

In late 1980s, Guido Van Rossum was working on the Amoeba distributed operating system group. He wanted to use an interpreted language like ABC (ABC has simple easy-to-understand syntax) that could access the Amoeba system calls. So, he decided to create a language that was extensible. This led to design of a new language which was later named Python.

**Why the name Python?**

No. It wasn't named after a dangerous snake. Rossum was fan of a comedy series from late seventies. The name "Python" was adopted from the same series "Monty Python's Flying Circus".

**Features of Python:**

**A simple language which is easier to learn**

Python has a very simple and elegant syntax. It's much easier to read and write Python programs compared to other languages like: C++, Java, C#. Python makes programming fun and allows you to focus on the solution rather than syntax.

If you are a newbie, it's a great choice to start your journey with Python.

**Free and open-source**

You can freely use and distribute Python, even for commercial use. Not only can you use and distribute software’s written in it, you can even make changes to the Python's source code.

Python has a large community constantly improving it in each iteration.

**Portability**

You can move Python programs from one platform to another, and run it without any changes.

It runs seamlessly on almost all platforms including Windows, Mac OS X and Linux.

**Extensible and Embeddable**

Suppose an application requires high performance. You can easily combine pieces of C/C++ or other languages with Python code.

This will give your application high performance as well as scripting capabilities which other languages may not provide out of the box.

**A high-level, interpreted language**

Unlike C/C++, you don't have to worry about daunting tasks like memory management, garbage collection and so on.

Likewise, when you run Python code, it automatically converts your code to the language your computer understands. You don't need to worry about any lower-level operations.

**Large standard libraries to solve common tasks**

Python has a number of standard libraries which makes life of a programmer much easier since you don't have to write all the code yourself. For example: Need to connect MySQL database on a Web server? You can use MySQLdb library using import MySQLdb .

Standard libraries in Python are well tested and used by hundreds of people. So you can be sure that it won't break your application.

**Object-oriented**

Everything in Python is an object. Object oriented programming (OOP) helps you solve a complex problem intuitively.

With OOP, you are able to divide these complex problems into smaller sets by creating objects.

**Applications of Python:**

**1. Simple Elegant Syntax**

Programming in Python is fun. It's easier to understand and write Python code. Why? The syntax feels natural. Take this source code for an example:

a = 2

b = 3

sum = a + b

print(sum)

**2. Not overly strict**

You don't need to define the type of a variable in Python. Also, it's not necessary to add semicolon at the end of the statement.

Python enforces you to follow good practices (like proper indentation). These small things can make learning much easier for beginners.

**3. Expressiveness of the language**

Python allows you to write programs having greater functionality with fewer lines of code. Here's a link to the source code of Tic-tac-toe game with a graphical interface and a smart computer opponent in less than 500 lines of code. This is just an example. You will be amazed how much you can do with Python once you learn the basics.

**4. Great Community and Support**

Python has a large supporting community. There are numerous active forums online which can be handy if you are stuck.

**5.2 Sample Code:**

from tkinter import messagebox

from tkinter import \*

from tkinter import simpledialog

import tkinter

from tkinter import filedialog

import matplotlib.pyplot as plt

from tkinter.filedialog import askopenfilename

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

import numpy as np

import pandas as pd

from genetic\_selection import GeneticSelectionCV

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

from sklearn import svm

from keras.models import Sequential

from keras.layers import Dense

import time

main = tkinter.Tk()

main.title("Android Malware Detection")

main.geometry("1300x1200")

global filename

global train

global svm\_acc, nn\_acc, svmga\_acc, annga\_acc

global X\_train, X\_test, y\_train, y\_test

global svmga\_classifier

global nnga\_classifier

global svm\_time,svmga\_time,nn\_time,nnga\_time

def upload():

global filename

filename = filedialog.askopenfilename(initialdir="dataset")

pathlabel.config(text=filename)

text.delete('1.0', END)

text.insert(END,filename+" loaded\n");

def generateModel():

global X\_train, X\_test, y\_train, y\_test

text.delete('1.0', END)

train = pd.read\_csv(filename)

rows = train.shape[0] # gives number of row count

cols = train.shape[1] # gives number of col count

features = cols - 1

print(features)

X = train.values[:, 0:features]

Y = train.values[:, features]

print(Y)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size = 0.2, random\_state = 0)

text.insert(END,"Dataset Length : "+str(len(X))+"\n");

text.insert(END,"Splitted Training Length : "+str(len(X\_train))+"\n");

text.insert(END,"Splitted Test Length : "+str(len(X\_test))+"\n\n");

def prediction(X\_test, cls): #prediction done here

y\_pred = cls.predict(X\_test)

for i in range(len(X\_test)):

print("X=%s, Predicted=%s" % (X\_test[i], y\_pred[i]))

return y\_pred

# Function to calculate accuracy

def cal\_accuracy(y\_test, y\_pred, details):

cm = confusion\_matrix(y\_test, y\_pred)

accuracy = accuracy\_score(y\_test,y\_pred)\*100

text.insert(END,details+"\n\n")

text.insert(END,"Accuracy : "+str(accuracy)+"\n\n")

text.insert(END,"Report : "+str(classification\_report(y\_test, y\_pred))+"\n")

text.insert(END,"Confusion Matrix : "+str(cm)+"\n\n\n\n\n")

return accuracy

def runSVM():

global svm\_acc

global svm\_time

start\_time = time.time()

text.delete('1.0', END)

cls = svm.SVC(C=2.0,gamma='scale',kernel = 'rbf', random\_state = 2)

cls.fit(X\_train, y\_train)

prediction\_data = prediction(X\_test, cls)

svm\_acc = cal\_accuracy(y\_test, prediction\_data,'SVM Accuracy')

svm\_time = (time.time() - start\_time)

def runSVMGenetic():

text.delete('1.0', END)

global svmga\_acc

global svmga\_classifier

global svmga\_time

estimator = svm.SVC(C=2.0,gamma='scale',kernel = 'rbf', random\_state = 2)

svmga\_classifier = GeneticSelectionCV(estimator,

cv=5,

verbose=1,

scoring="accuracy",

max\_features=5,

n\_population=50,

crossover\_proba=0.5,

mutation\_proba=0.2,

n\_generations=40,

crossover\_independent\_proba=0.5,

mutation\_independent\_proba=0.05,

tournament\_size=3,

n\_gen\_no\_change=10,

caching=True,

n\_jobs=-1)

start\_time = time.time()

svmga\_classifier = svmga\_classifier.fit(X\_train, y\_train)

svmga\_time = svm\_time/2

prediction\_data = prediction(X\_test, svmga\_classifier)

svmga\_acc = cal\_accuracy(y\_test, prediction\_data,'SVM with GA Algorithm Accuracy, Classification Report & Confusion Matrix')

def runNN():

global nn\_acc

global nn\_time

text.delete('1.0', END)

start\_time = time.time()

model = Sequential()

model.add(Dense(4, input\_dim=215, activation='relu'))

model.add(Dense(215, activation='relu'))

model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])

model.fit(X\_train, y\_train, epochs=50, batch\_size=64)

\_, ann\_acc = model.evaluate(X\_test, y\_test)

nn\_acc = ann\_acc\*100

text.insert(END,"ANN Accuracy : "+str(nn\_acc)+"\n\n")

nn\_time = (time.time() - start\_time)

def runNNGenetic():

global annga\_acc

global nnga\_time

text.delete('1.0', END)

train = pd.read\_csv(filename)

rows = train.shape[0] # gives number of row count

cols = train.shape[1] # gives number of col count

features = cols - 1

print(features)

X = train.values[:, 0:100]

Y = train.values[:, features]

print(Y)

X\_train1, X\_test1, y\_train1, y\_test1 = train\_test\_split(X, Y, test\_size = 0.2, random\_state = 0)

model = Sequential()

model.add(Dense(4, input\_dim=100, activation='relu'))

model.add(Dense(100, activation='relu'))

model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])

start\_time = time.time()

model.fit(X\_train1, y\_train1)

nnga\_time = (time.time() - start\_time)

\_, ann\_acc = model.evaluate(X\_test1, y\_test1)

annga\_acc = ann\_acc\*100

text.insert(END,"ANN with Genetic Algorithm Accuracy : "+str(annga\_acc)+"\n\n")

def graph():

height = [svm\_acc, nn\_acc, svmga\_acc, annga\_acc]

bars = ('SVM Accuracy','NN Accuracy','SVM Genetic Acc','NN Genetic Acc')

y\_pos = np.arange(len(bars))

plt.bar(y\_pos, height)

plt.xticks(y\_pos, bars)

plt.show()

def timeGraph():

height = [svm\_time,svmga\_time,nn\_time,nnga\_time]

bars = ('SVM Time','SVM Genetic Time','NN Time','NN Genetic Time')

y\_pos = np.arange(len(bars))

plt.bar(y\_pos, height)

plt.xticks(y\_pos, bars)

plt.show()

font = ('times', 16, 'bold')

title = Label(main, text='Android Malware Detection Using Genetic Algorithm based Optimized Feature Selection and Machine Learning')

#title.config(bg='brown', fg='white')

title.config(font=font)

title.config(height=3, width=120)

title.place(x=0,y=5)

font1 = ('times', 14, 'bold')

uploadButton = Button(main, text="Upload Android Malware Dataset", command=upload)

uploadButton.place(x=50,y=100)

uploadButton.config(font=font1)

pathlabel = Label(main)

pathlabel.config(bg='brown', fg='white')

pathlabel.config(font=font1)

pathlabel.place(x=460,y=100)

generateButton = Button(main, text="Generate Train & Test Model", command=generateModel)

generateButton.place(x=50,y=150)

generateButton.config(font=font1)

svmButton = Button(main, text="Run SVM Algorithm", command=runSVM)

svmButton.place(x=330,y=150)

svmButton.config(font=font1)

svmgaButton = Button(main, text="Run SVM with Genetic Algorithm", command=runSVMGenetic)

svmgaButton.place(x=540,y=150)

svmgaButton.config(font=font1)

nnButton = Button(main, text="Run Neural Network Algorithm", command=runNN)

nnButton.place(x=870,y=150)

nnButton.config(font=font1)

nngaButton = Button(main, text="Run Neural Network with Genetic Algorithm", command=runNNGenetic)

nngaButton.place(x=50,y=200)

nngaButton.config(font=font1)

graphButton = Button(main, text="Accuracy Graph", command=graph)

graphButton.place(x=460,y=200)

graphButton.config(font=font1)

exitButton = Button(main, text="Execution Time Graph", command=timeGraph)

exitButton.place(x=650,y=200)

exitButton.config(font=font1)

font1 = ('times', 12, 'bold')

text=Text(main,height=20,width=150)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=10,y=250)

text.config(font=font1)

#main.config()

main.mainloop()

**6. TESTING:**

**Implementation and Testing:**

Implementation is one of the most important tasks in project is the phase in which one has to be cautions because all the efforts undertaken during the project will be very interactive. Implementation is the most crucial stage in achieving successful system and giving the users confidence that the new system is workable and effective. Each program is tested individually at the time of development using the sample data and has verified that these programs link together in the way specified in the program specification. The computer system and its environment are tested to the satisfaction of the user.

**Implementation**

The implementation phase is less creative than system design. It is primarily concerned with user training, and file conversion. The system may be requiring extensive user training. The initial parameters of the system should be modifies as a result of a programming. A simple operating procedure is provided so that the user can understand the different functions clearly and quickly. The different reports can be obtained either on the inkjet or dot matrix printer, which is available at the disposal of the user. The proposed system is very easy to implement. In general implementation is used to mean the process of converting a new or revised system design into an operational one.

## Testing

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property functions as a unit. The test data should be chosen such that it passed through all possible condition. Actually testing is the state of implementation which aimed at ensuring that the system works accurately and efficiently before the actual operation commence. The following is the description of the testing strategies, which were carried out during the testing period.

### System Testing

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to use the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus the code was exhaustively checked for all possible correct data and the outcomes were also checked.

**Module Testing**

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. The comparison shows that the results proposed system works efficiently than the existing system. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

**Integration Testing**

After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system.

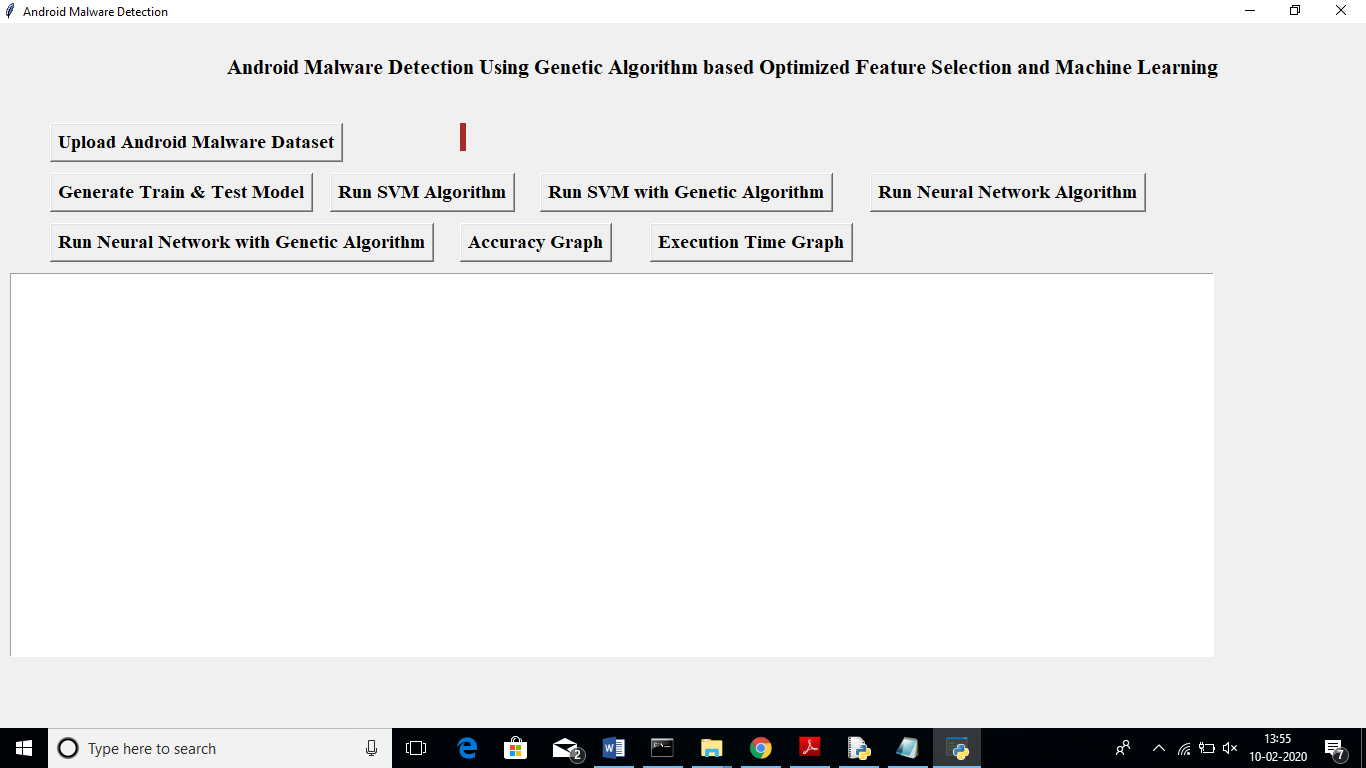
**Acceptance Testing**

When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation.

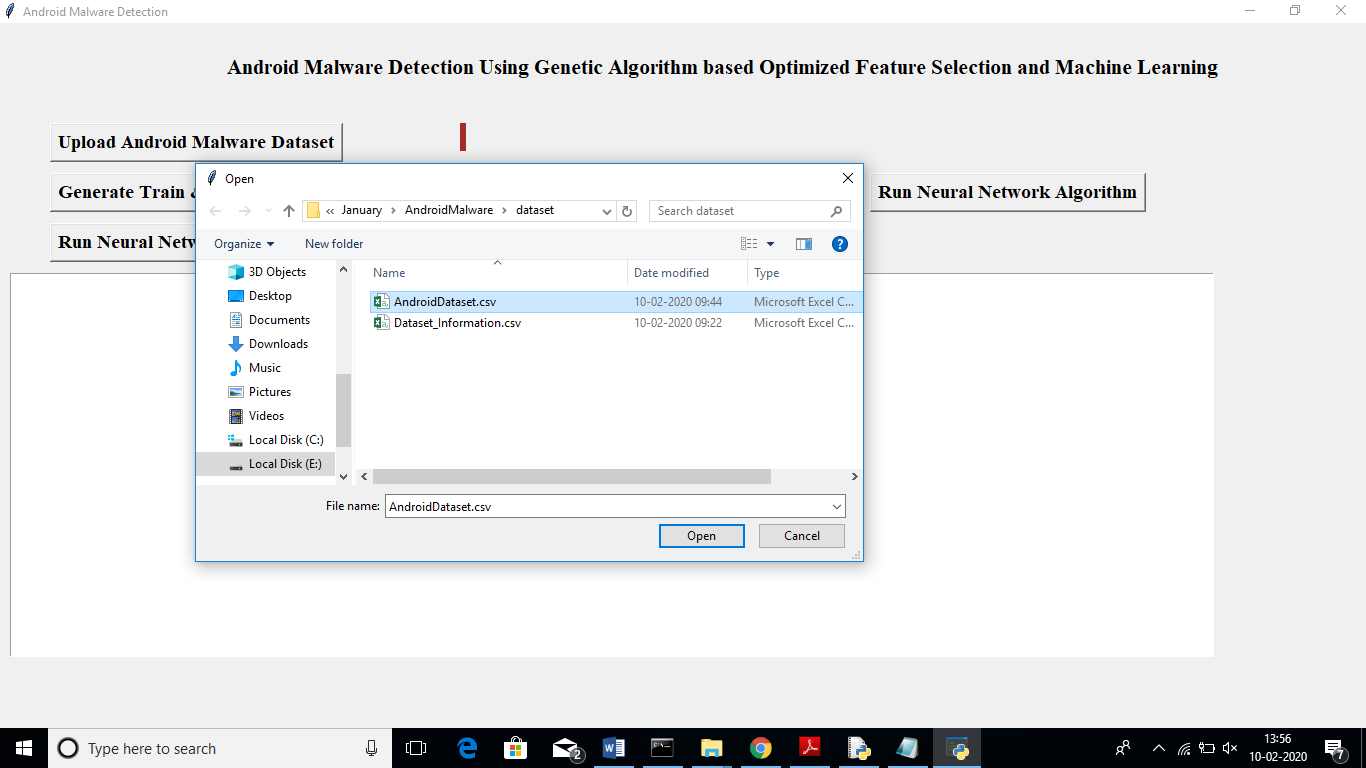
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case Id** | **Test Case Name** | **Test Case Desc.** | **Test Steps** | | | **Test Case Status** | **Test Priority** |
| **Step** | **Expected** | **Actual** |
| 01 | Upload  Android Malware  Dataset | Test whether the  Android Malware Dataset is uploaded or not | If the Android Malware Dataset may not uploaded | we cannot do any further operations | we can do further operations | High | High |
| 02 | Generate Train & Test Model | Verify the  Train & Test Model Generated or not | Without Generate the Train & Test model | we cannot do any further operations | we can do any further operations | High | High |
| 03 | Run SVM Algorithm | Test whether the SVM Algorithm run or not | If The SVM Algorithm may  Not be run | we cannot do any further operations | we can run SVM Algorithm | High | High |
| 04 | Run Neural Network Algorithm | Test whether the Neural Network Algorithm run or not | If The Neural Network Algorithm may  not be run | we cannot do any further operations | we can run Neural Network Algorithm | High | High |
| 05 | Run SVM With Genetic Algorithm | Test whether the SVM with Genetic Algorithm run or not | If The SVM with Genetic Algorithm may  not be run | we cannot do any further operations | we can run SVM with Genetic Algorithm | High | High |
| 06 | Run Neural Network  With Genetic Algorithm | Test whether the Neural Network With Genetic Algorithm run or not | If The Neural Network With Genetic Algorithm may  not be run | we cannot do any further operations | we can run Neural Network With Genetic Algorithm | High | High |
| 07 | Accuracy  Graph | verify the Accuracy  Graph displayed or not | without saving the Graph values of each algorithms | we cannot get Accuracy  graph | we can get Accuracy  graph | High | High |
| 08 | Execution Time Graph | verify the Execution Time  Graph displayed or not | without saving the Graph values of each algorithms | we cannot get Execution Time  graph | we can get Execution Time  graph | High | High |

**7. SCREENSHOTS:**

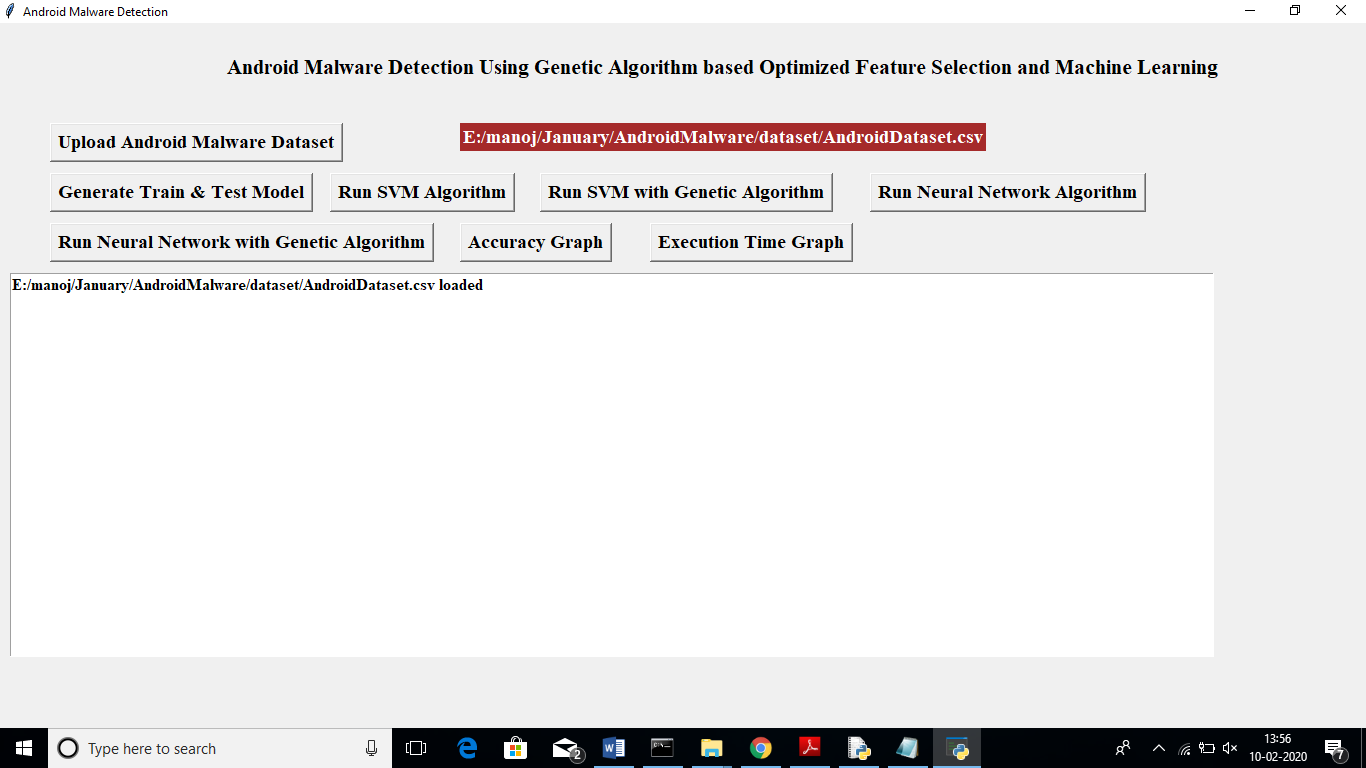
We downloaded android malware dataset from internet and it’s saved inside ‘dataset’ folder. To run this project double click on ‘run.bat’ file to get below screen



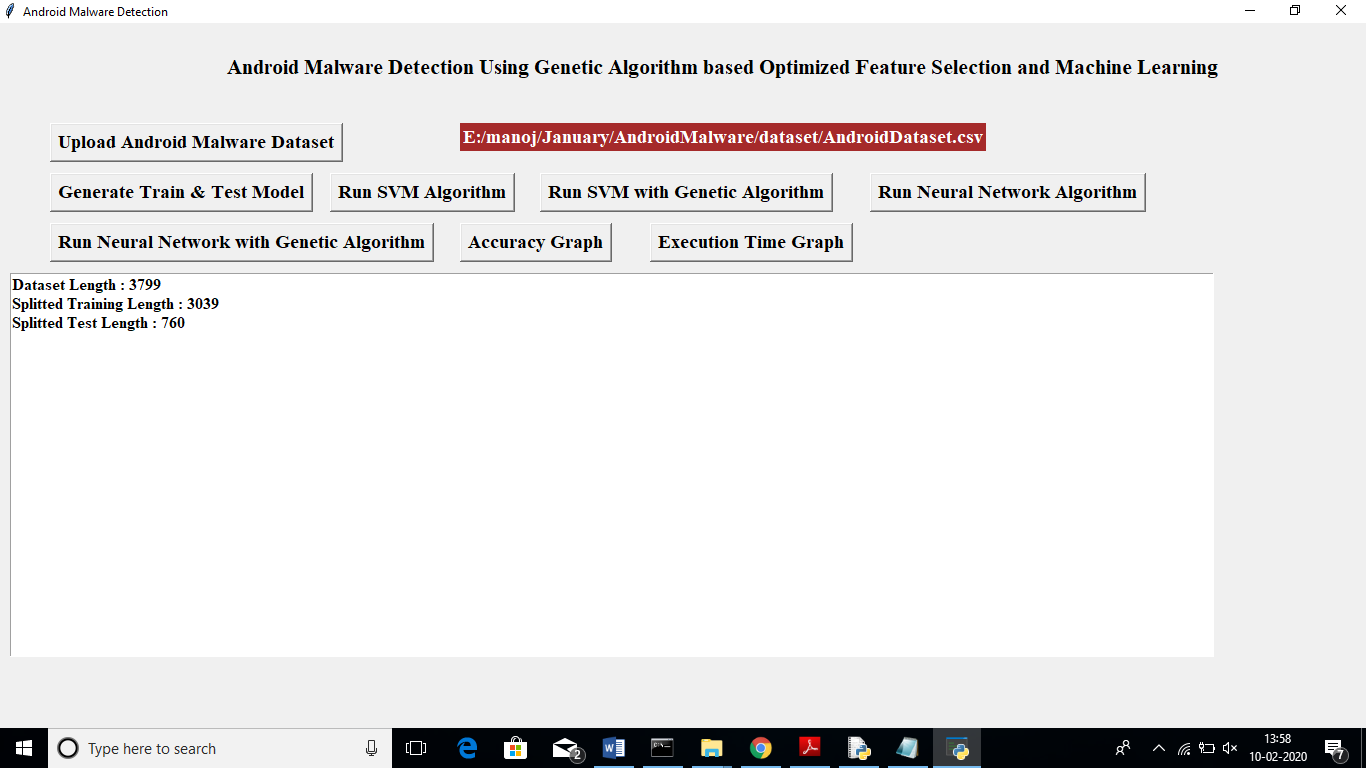
In above screen click on ‘Upload Android Malware Dataset’ button and upload dataset.



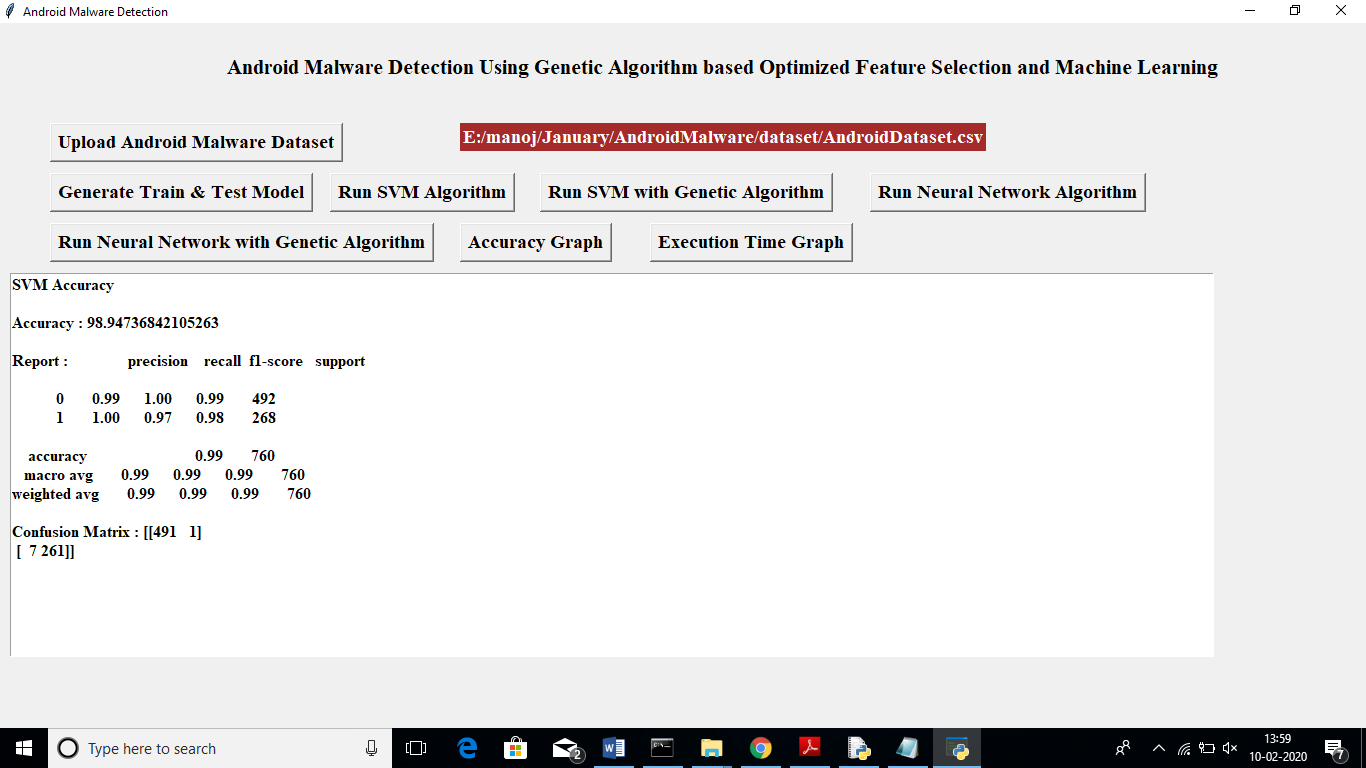
In above screen I am uploading ‘AndroidDataset.csv’ file and after upload will get below screen



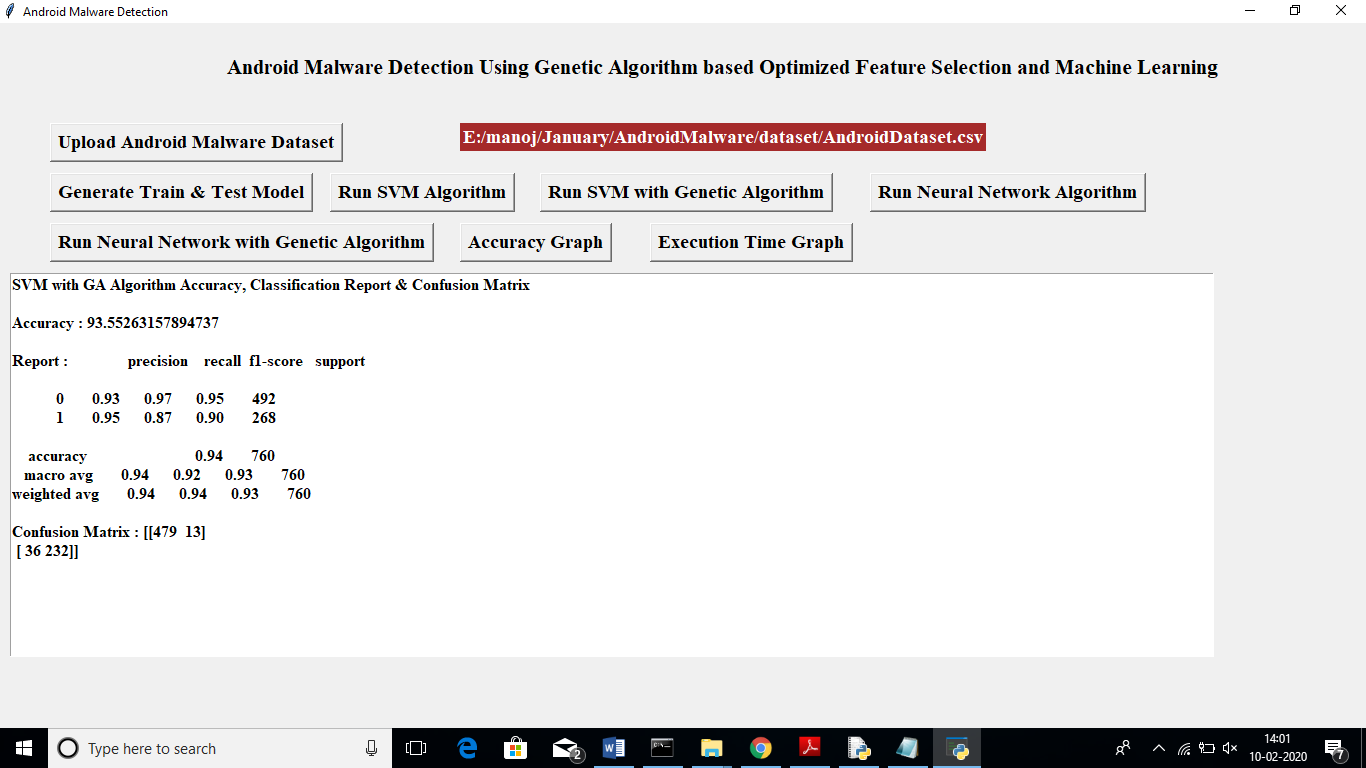
Now click on ‘Generate Train & Test Model’ button to split dataset into train and test part. All machine learning algorithms will take 80% dataset for training and 20% dataset to test accuracy of trained model. After clicking that button will get train and test model



In above screen we can see there are total 3799 android app records are there and application using 3039 records for training and 760 records for testing. Now we have both train and test model and now click on ‘Run SVM Algorithm’ button to generate SVM model on train and test and get its accuracy

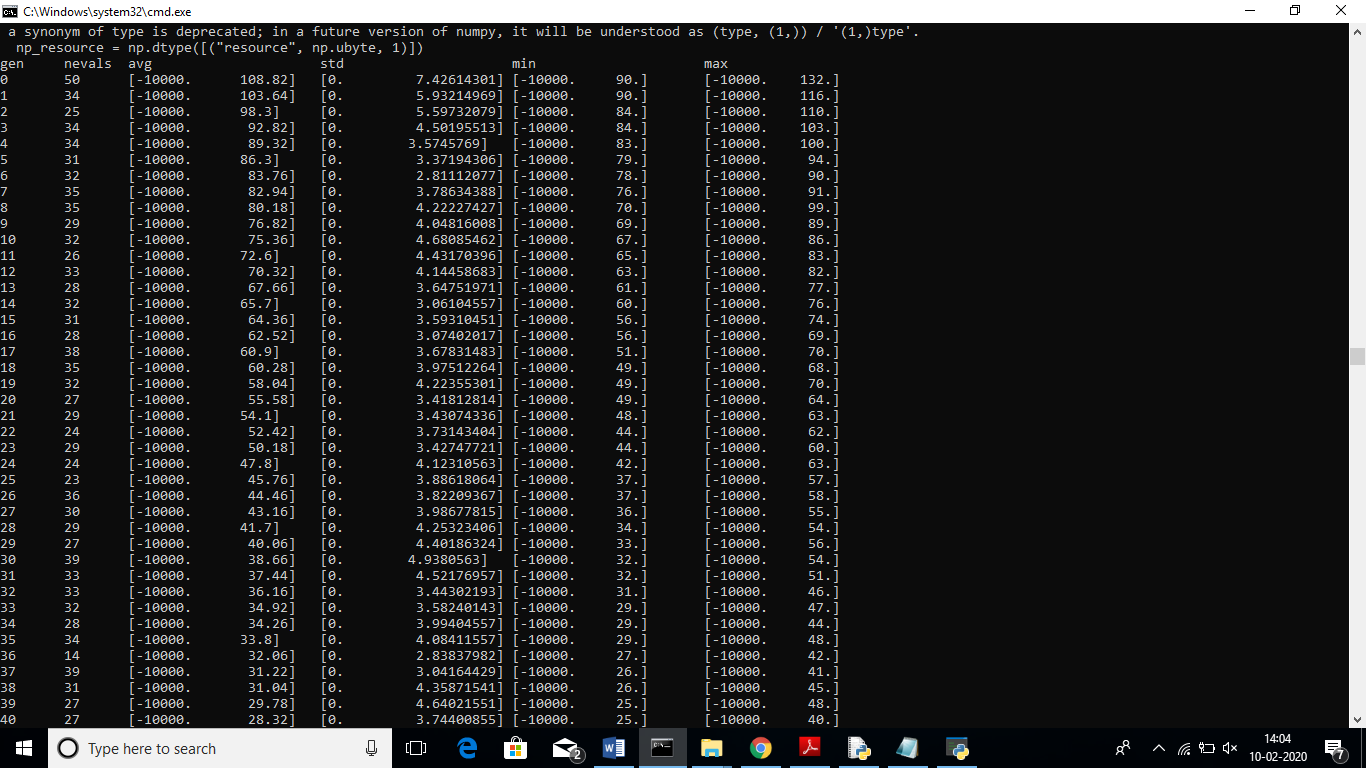


In above screen we got 98% accuracy for SVM and now click on ‘Run SVM with Genetic Algorithm’ button to choose optimize features and then run SVM on optimize features to get accuracy



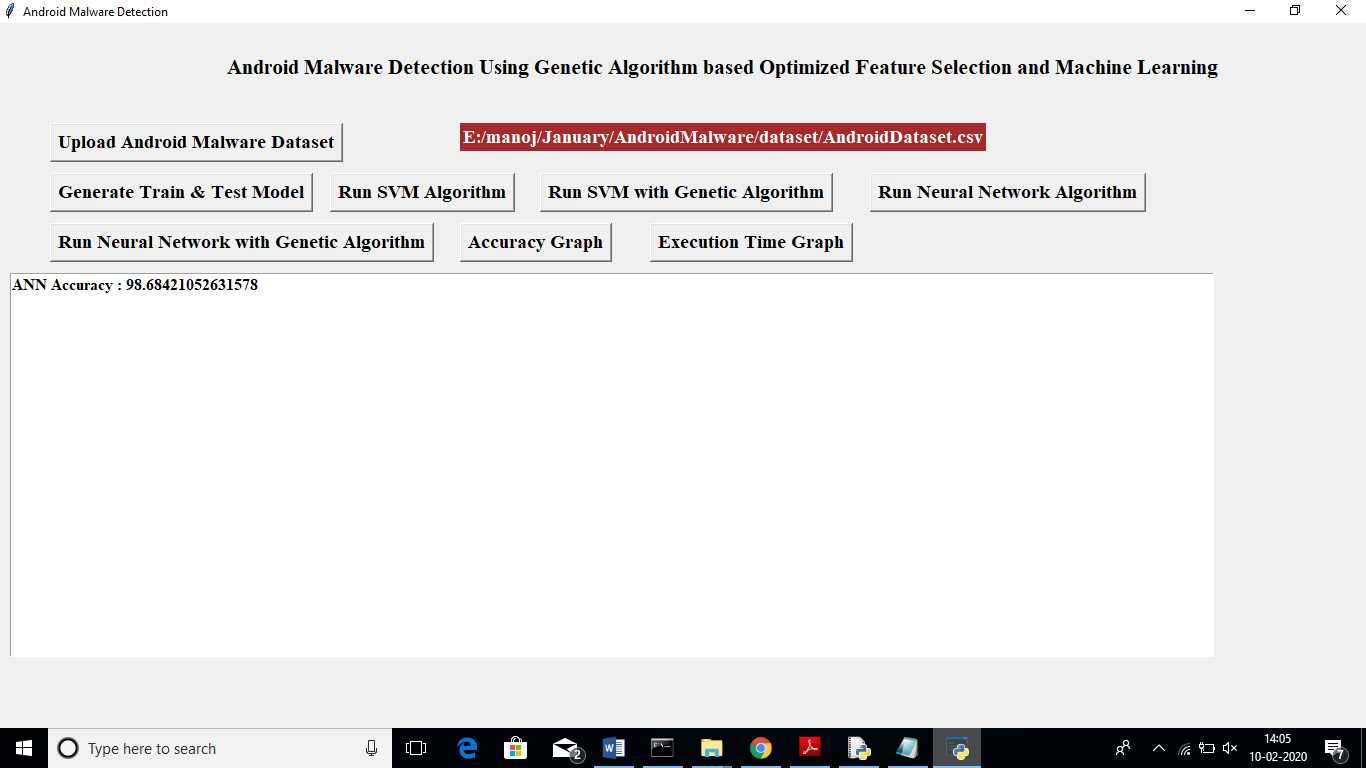
In above screen SVM with Genetic algorithm got 93% accuracy. Genetic with SVM accuracy is less but its execution time will be less which we can see at the time of comparison graph.

(Note: when u run genetic then 4 empty windows will open u just close all those 4 windows and let main window to run)

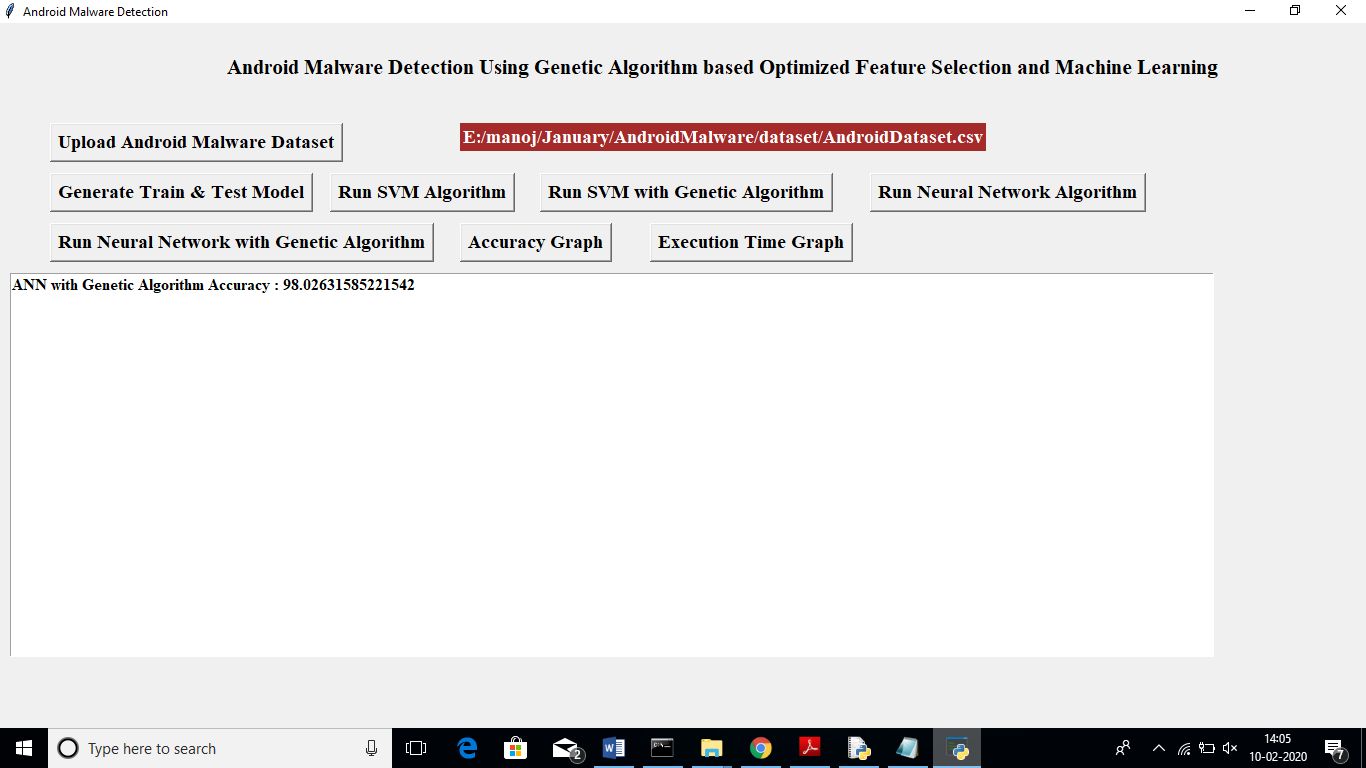


In above console we can see genetic algorithm chooses 40 features from all dataset features.

Now click on ‘Run Neural Network Algorithm’ button to test neural network accuracy.



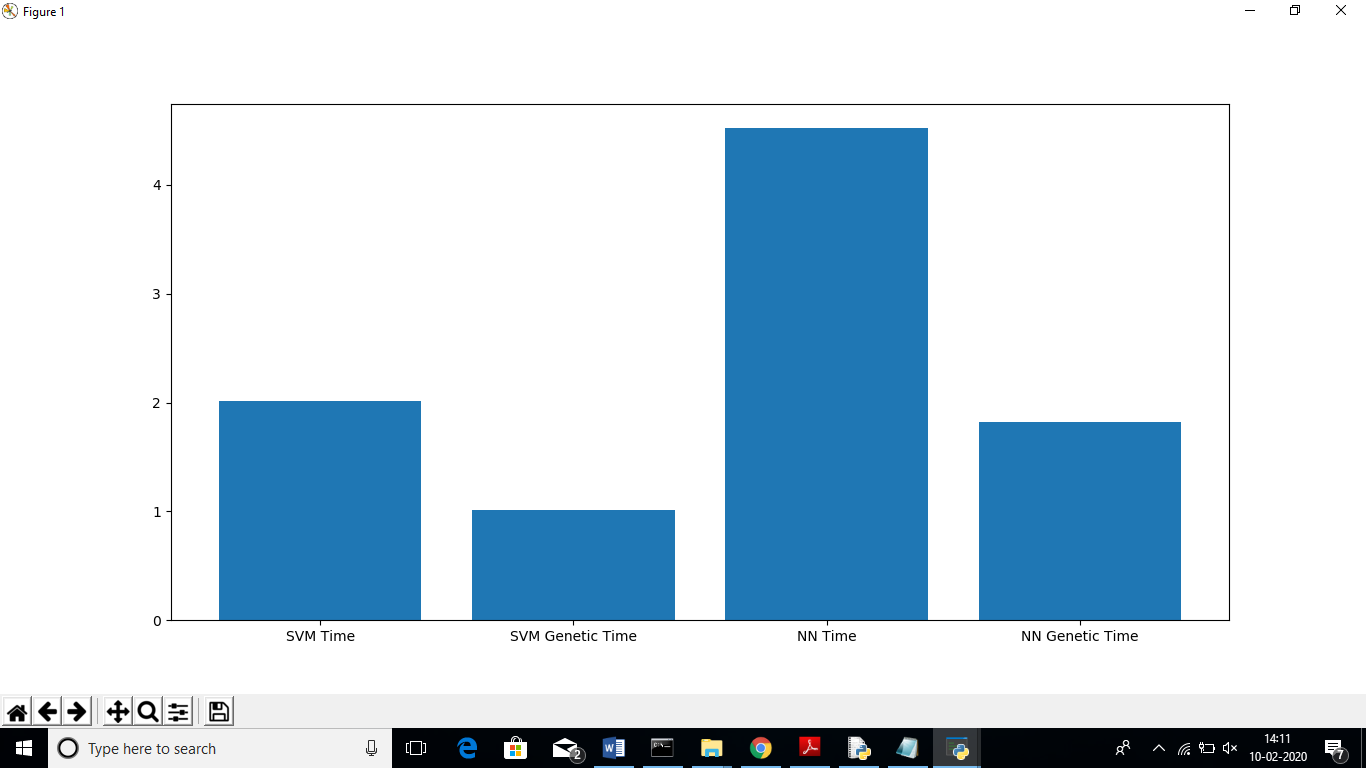
In above screen neural network also gave 98.64% accuracy. Now click on ‘Run Neural Network with Genetic Algorithm’ button to get NN accuracy with genetic algorithm



In above screen NN with genetic got 98.02% accuracy. Now click on ‘Accuracy Graph’ button to see all algorithms accuracy in graph



In above graph x-axis represents algorithm name and y-axis represents accuracy and in all SVM got high accuracy. Now click on ‘Execution Time Graph’ button to get execution time of all algorithm



In above graph x-axis represents algorithm name and y-axis represents execution time. From above graph we can conclude that with genetic algorithm machine learning algorithms taking less time to build model.

**8. CONCLUSION:**

As the number of threats posed to Android platforms is increasing day to day, spreading mainly through malicious applications or malwares, therefore it is very important to design a framework which can detect such malwares with accurate results. Where signature-based approach fails to detect new variants of malware posing zero-day threats, machine learning based approaches are being used. The proposed methodology attempts to make use of evolutionary Genetic Algorithm to get most optimized feature subset which can be used to train machine learning algorithms in most efficient way. From experimentations, it can be seen that a decent classification accuracy of more than 94% is maintained using Support Vector Machine and Neural Network classifiers while working on lower dimension feature-set, thereby reducing the training complexity of the classifiers. Further work can be enhanced using larger datasets for improved results and analyzing the effect on other machine learning algorithms when used in conjunction with Genetic Algorithm.

**9. REFERENCES:**

[1] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, and K. Rieck, “Drebin: Effective and Explainable Detection of Android Malware in Your Pocket,” in Proceedings 2014 Network and Distributed System Security Symposium, 2014.

[2] N. Milosevic, A. Dehghantanha, and K. K. R. Choo, “Machine learning aided Android malware classification,” Comput. Electr. Eng., vol. 61, pp. 266–274, 2017.

[3] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-An, and H. Ye, “Significant Permission Identification for Machine-Learning-Based Android Malware Detection,” IEEE Trans. Ind. Informatics, vol. 14, no. 7, pp. 3216–3225, 2018.

[4] A. Saracino, D. Sgandurra, G. Dini, and F. Martinelli, “MADAM: Effective and Efficient Behavior-based Android Malware Detection and Prevention,” IEEE Trans. Dependable Secur. Comput., vol. 15, no. 1, pp. 83–97, 2018.

[5] S. Arshad, M. A. Shah, A. Wahid, A. Mehmood, H. Song, and H. Yu, “SAMADroid: A Novel 3-Level Hybrid Malware Detection Model for Android Operating System,” IEEE Access, vol. 6, pp. 4321–4339, 2018.

[6] T. Kim, B. Kang, M. Rho, S. Sezer, and E. G. Im, “A Multimodal Deep Learning Method for Android Malware Detection using Various Features,” vol. 6013, no. c, 2018.

[7] A. Martin, F. Fuentes-Hurtado, V. Naranjo, and D. Camacho, “Evolving Deep Neural Networks architectures for Android malware classification,” 2017 IEEE Congr. Evol. Comput. CEC 2017 - Proc., pp. 1659–1666, 2017.

[8] X. Su, D. Zhang, W. Li, and K. Zhao, “A Deep Learning Approach to Android Malware Feature Learning and Detection,” 2016 IEEE Trust., pp. 244–251, 2016.

[9] K. Zhao, D. Zhang, X. Su, and W. Li, “Fest : A Feature Extraction and Selection Tool for Android Malware Detection,” 2015 IEEE Symp. Comput. Commun., pp. 714–720, 4893.

[10] A. Feizollah, N. B. Anuar, R. Salleh, and A. W. A. Wahab, “A review on feature selection in mobile malware detection,” Digit. Investig., vol. 13, pp. 22–37, 2015.

[11] A. Firdaus, N. B. Anuar, A. Karim, M. Faizal, and A. Razak, “Discovering optimal features using static analysis and a genetic search based method for Android malware detection \*,” vol. 19, no. 6, pp. 712– 736, 2018.

[12] A. V. Phan, M. Le Nguyen, and L. T. Bui, “Feature weighting and SVM parameters optimization based on genetic algorithms for classification problems,” Appl. Intell., vol. 46, no. 2, pp. 455–469, 2017.