A

Mini Project Report on

AN EFFICIENT ANDROID MALWARE DETECTION USING GA, SVM AND ANN.

Submitted for partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

by

MEDIKONDA NAVEEN BABU	**18K81A0593**
NUCHU ABHIJEET PRAKASH YADAV	**18K81A05A0**
PASUPULETI KRISHNA TEJASVI	**18K81A05A3**
RAPOLE MANISH	**18K81A05A7**

Under the Guidance of

Dr. K. SRINIVAS

ASSISTANT PROFESSOR

DEPARTMENT OF CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

St. MARTIN'S ENGINEERING COLLEGE

An Autonomous Institute

NBA & NAAC A+ Accredited

Dhulapally, Secunderabad - 500 100

JANUARY - 2022

St. MARTIN'S ENGINEERING COLLEGE

An Autonomous Institute

NBA & NAAC A+ Accredited

Dhulapally, Secunderabad - 500 100

ww.smec.ac.in

CERTIFICATE

This is to certify that the project entitled **"An efficient android malware detection using GA, SVM and ANN"** is being submitted by **Medikonda Naveen Babu 18K81A0593, Nuchu Abhijeet Prakash Yadav 18K81A05A0, Pasupuleti Krishna Tejasvi 18K81A05A3, Manish Rapole 18K81A05A7**, in fulfilment of the requirement for the award of degree of **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE & ENGINEERING** is recorded of bonafide work carried out by them. The resultembodied in this report have been verified and found satisfactory.

Guide	**Head of the Department**
Dr. K. SRINIVAS	**Dr. M. NARAYANAN**
ASSISTANT PROFESSOR	**PROFESSOR & HEAD**
DEPARTMENT OF CSE	**DEPARTMENT OF CSE**

Internal Examiner **External Examiner**

Place:

Date:

St. MARTIN'S ENGINEERING COLLEGE

An Autonomous Institute

NBA & NAAC A+ Accredited

Dhulapally, Secunderabad - 500 100

www.smec.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, the students of '**Bachelor of Technology in Department of Computer Science and Engineering'**, session: 2018 - 2022**, St. Martin's Engineering College, Dhulapally, Kompally, Secunderabad,** hereby declare that the work presented in this Mini Project Work entitled **"An efficient Android malware detection using GA, SVM and ANN"** is the outcome of our own bonafide work and is correct to the best of our knowledge and this work has been undertaken taking care of Engineering Ethics. This result embodied in this project report has not been submitted in any university for award of any degree.

Mr. MEDIKONDA NAVEEN BABU	**18K81A0593**
Mr. NUCHU ABHIJEET PRAKASH YADAV	**18K81A05A0**
Ms. PASUPULETI KRISHNA TEJASVI	**18K81A05A3**
Mr. RAPOLE MANISH	**18K81A05A7**

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose encouragement and guidance have crowded our efforts with success.

We extend our deep sense of gratitude to Principal, **Dr. P. SANTOSH KUMAR PATRA**, St. Martin's Engineering College Dhulapally, for permitting us to undertake this project.

We are also thankful to **Dr. M. NARAYANAN**, Head of the Department, Computer Science and Engineering, St. Martin's Engineering College, Dhulapally, Secunderabad. for his support and guidance throughout our project as well as Project Coordinator **Dr. K. SRINIVAS**, Associate Professor, Computer Science and Engineering department for his valuable support.

We would like to express our sincere gratitude and indebtedness to our project supervisor **Dr. K. SRINIVAS**, Assistant Professor, Department of Computer Science and Engineering, St. Martins Engineering College, Dhulapally, for his support and guidance throughout our project.

Finally, we express thanks to all those who have helped us successfully completing this project. Furthermore, we would like to thank our family and friends for their moral support and encouragement. We express thanks to all those who have helped us in successfully completing the project.

Mr. MEDIKONDA NAVEEN BABU	**18K81A0593**
Mr. NUCHU ABHIJEET PRAKASH YADAV	**18K81A05A0**
Ms. PASUPULETI KRISHNA TEJASVI	**18K81A05A3**
Mr. RAPOLE MANISH	**18K81A05A7**

ABSTRACT

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 project 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.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS AND DEFINITIONS

S.NO	ACRONYM	DEFINITION
01	ANN	Artificial neural network
02	DFD	Data Flow Diagram
03	DNN	Data Neural Network
04	GA	Genetic Algorithm
05	MADAM	Mutli-Level Anomaly Detector for Android Malware
06	NIS	Network Interface Card
07	PDR	Performance and Development Review
08	RTM	Requirements Traceability Matrix
09	SDLC	Software Development Life Cycle
10	SigPID	Significant Permission Identification
11	SVM	Support Vector Machine
12	UML	Unified Modelling Language

1. INTRODUCTION

Android Apps are freely available on Google Play store, the official Android app store as well as third-party app stores for users to download. It is an open-source nature and popularity; malware writers are increasingly focusing on developing malicious applications for Android operating system. Despite various attempts by Google Play store 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 a multi-level analysis framework where behaviour of Android Apps is captured up to 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

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 marketplaces. This gives malware developers an opportunity to put repackaged malicious applications in third-party app stores and attack Android devices. Many malware analysis and detection systems have been developed which uses static analysis, dynamic analysis, or hybrid analysis to keep Android devices secure from malware [1].

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 [2].

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 [3].

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 [4].

Deep Neural Networks (DNN) has become a powerful, widely used, and successful mechanism to solve problems of different nature and varying complexity.

Their ability to build models adapted to complex non-linear problems has 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 behavior of the application as exhaustively as possible [5].

A novel approach is proposed to detect malware in a hybrid and generic way, especiallyfor mobile malware in Android devices. By comparing the patterns reflected by the above individual and sequential of malware and benign apps with each other, we buildup a malicious pattern set and a normal pattern set that are used for malware detectionand benign app judgment. When we need to detect an unknown app, we use a dynamicmethod to collect its runtime system calling data in terms of both individual calls and sequential system calls with different depths. Then we extract the target patterns of theunknown app from its runtime system calling data [6].

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 the static and dynamic analysis. In this paper, we present two machinelearning-aided approaches for static analysis of Android malware [7].

The growing amount and diversity of Android malware have significantly weakened the effectiveness of the conventional defence 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 [8].

The contributions include: description of the development of a deep-learning-based Android malware detection engine (DroidDetector)that has been put online for user testing and can automatically detect whether an app is a malware or not. In this software, 20 000 apps from the Google Play Store and collect 1760 malware from the well-known Contagio Community and Genome Project [9].

In particular, to overcome the drawbacks of previous malware-centric methods, a novel and feature-rich hybrid anti-malware system, called Andro-Dumpsys is proposed. An integrated anti-malware analysis system that considers both malware-centric information and malware creator-centric information for malware analysis. Using malware creator-centric information simplifies the process of malware analysis. Andro-Dumpsys facilitates the detection of new malware including malware variants and zero-day malware. This is further highlighted by experiments using real-world up-to-date malware samples [10].

MONET, an Android malware detection system that consists of a client module and a backend server module. The client module is a lightweight, in-device app for malware behaviour monitoring and signature generation using two novel interception techniques, while the backend the server module is responsible for malware signature detection. The system can accurately describe the behaviours of an app to detect and classify malware variants and defend against to obfuscation attacks. The main focus is on classifying malware based on their behaviour similarity. Moreover, it has low computational overhead and low demand for battery resources. Monet is Designed and implemented for a runtime behaviour signature that can represent both the logic structures and the run time behaviours of an app. Our runtime behaviour signature is effective to detect malware variants and transforming malware. It can perform a lightweight, in-device malware detection system, for Android devices [11].

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 theyare favorite targets of attackers [12].

The battle to mitigate Android malware has become more critical with the emergence of new strains incorporating increasingly sophisticated evasion techniques, in turn necessitating more advanced detection capabilities. Thus, a machine learning-based anapproach based on eigenspace analysis for Android malware detection using features derived from static analysis characterization of Android applications. The eigenfaces method measures the extent of similarities and differences amongst a set of faces in a database. It is based on the premise that every face is a linear combination of the basicset of faces called eigenfaces [13].

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 in their frequencies between malware and benign apps, because features which are frequently used in malware and rarely used in benign apps are moreimportant to distinguish malware from benign apps [14].

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 unprotected from novel malware. In this paper, we propose DREBIN, a lightweight method for the detection of Android malware that enables identifying malicious applications directly 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 application as possible [15].

3. SYSTEM DESIGN

3.1 UML DIAGRAMS

UML (Unified Modelling Language) is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems.

3.1.1 USECASE DIAGRAMS

A use case diagram is used to represent the dynamic behaviour of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application.

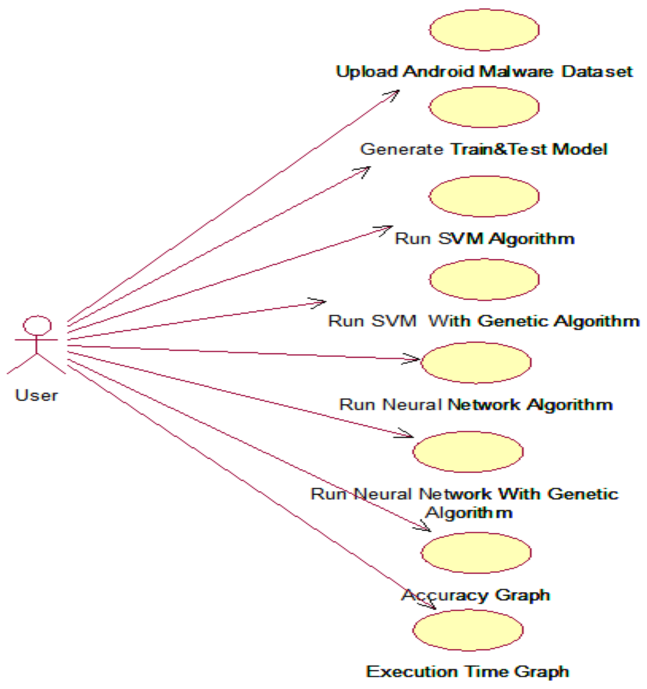

Upload Android Malware Dataset
Generate Train&Test Model
Run SVM Algorithm
Run SVM With Genetic Algorithm
User
Run Neural Network Algorithm
Run Neural Network With Genetic Algorithm
Accuracy Graph
Execution Time Graph

Fig.3.1: Use Case diagram for Android Malware Detection

Fig. 3.1 represents the use case diagram which refers to all the functionalitieswhich are provided to the user. There is only one end user, and he is functionalities.

3.1.2 CLASS DIAGRAM

It is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. Fig. 3.2 represents the Class Diagram, it refers to all the functions present in various classes, in our project there is only a single class.

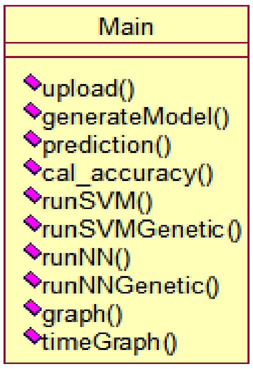
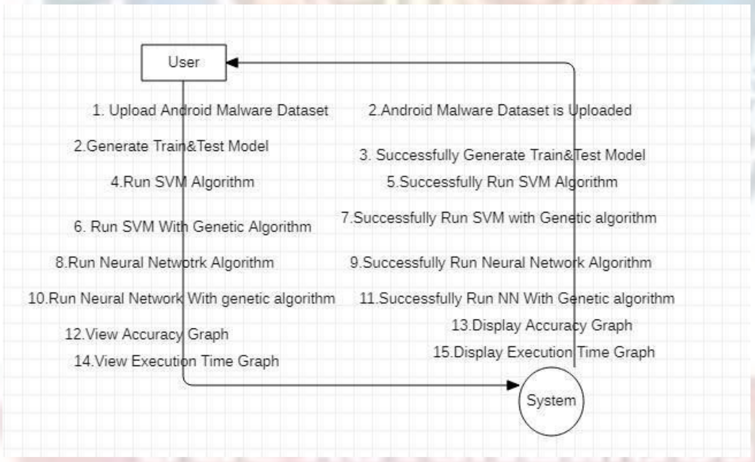


Fig.3.2: Class Diagram for Android Malware Detection

3.1.3 DATA FLOW DIAGRAM

Data flow diagram is a visual interpretation of information flow in any project or program. The data flow diagram is also called as data flow graph. In this project, the user contains provides the information by uploading dataset, this dataset is then sent for generating and testing the model, Fig. 3.3 represents the data flow graph.



User
1. Upload Android Malware Dataset
2.Android Malware Dataset is Uploaded
2.Generate Train&Test Model
3. Successfully Generate Train&Test Model
4.Run SVM Algorithm
5.Successfully Run SVM Algorithm
6. Run SVM With Genetic Algorithm
7.Successfully Run SVM with Genetic algorithm
8.Run Neural Netwotrk Algorithm
9.Successfully Run Neural Network Algorithm
10.Run Neural Network With genetic algorithm
11.Successfully Run NN With Genetic algorithm
12.View Accuracy Graph
13.Display Accuracy Graph
14.View Execution Time Graph
15.Display Execution Time Graph
System

Fig.3.3: Data Flow Diagram for Android Malware Detection

3.1.4 SEQUENCE DIAGRAM

Fig. 3.4 is sequence diagram 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. Sequence Diagram states the order which has to be followed.

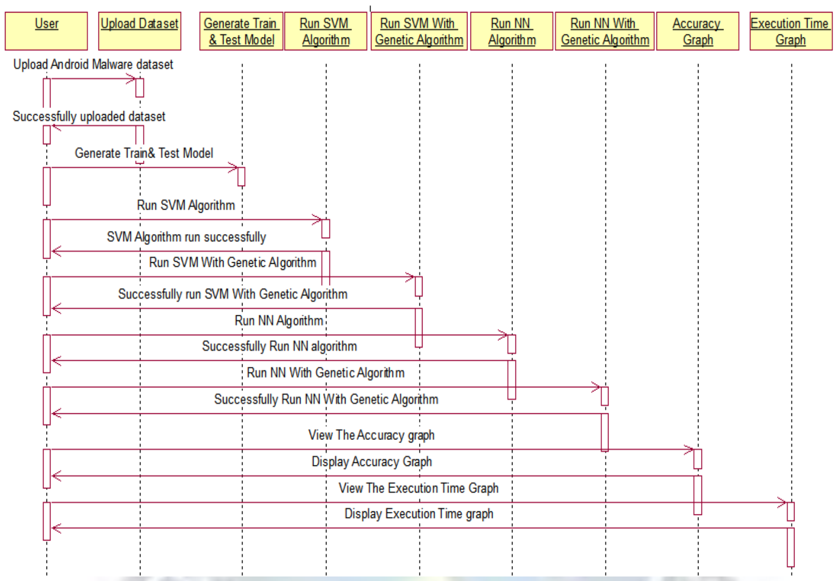

User
Upload Dataset
Generate Train & Test Model
Run SVM Algorithm
Run SVM With Genetic Algorithm
Run NN Algorithm
Run NN With Genetic Algorithm
Accuracy Graph
Execution Time Graph
Upload Android Malware dataset
Successfully uploaded dataset
Generate Train& Test Model
Run SVM Algorithm
SVM Algorithm run successfully
Run SVM With Genetic Algorithm
Successfully run SVM With Genetic Algorithm
Run NN Algorithm
Successfully Run NN algorithm
Run NN With Genetic Algorithm
Successfully Run NN With Genetic Algorithm
View The Accuracy graph
Display Accuracy Graph
View The Execution Time Graph
Display Execution Time graph

Fig.3.4: Sequence Diagram for Android Malware Detection

3.2 ARCHITECTURE

The Good ware APKs set and Malware APKs set combined called dataset are Reverse Engineered and the features are extracted by androguard tool, the discriminant features are selected by feature selection from generational genetic algorithm. The architecture diagram is given in Fig. 3.5. The accuracy is calculated before and after applying feature selection.

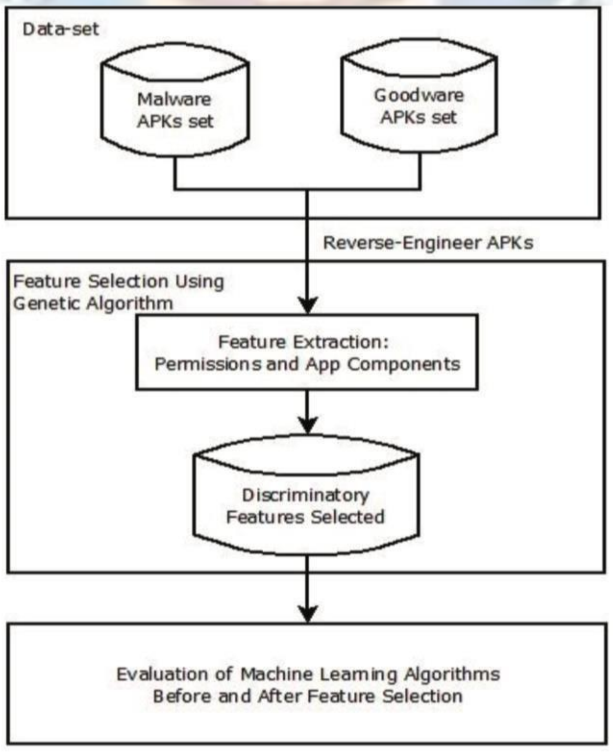

Data-set
Malware APKs set
Goodware APKs set
Reverse-Engineer APKs
Feature Selection Using Genetic Algorithm
Feature Extraction: Permissions and App Components
Discriminatory Features Selected
Evaluation of Machine Learning Algorithms Before and After Feature Selection

Fig.3.5: Architecture Diagram

3.3 REQUIRMENTS

3.3.1 MINIMUM HARDWARE TMREQUIREMENTS

Processor	-	Intel i5
RAM	-	4GB
Hard Disk	-	1 TB

3.3.2 MINIMUM SOFTWARE REQUIREMENTS

Operating System	-	Windows 11
Programming Language	-	Python (3.7)

3.4 PROJECT PLANNING AND PROCESS MODEL

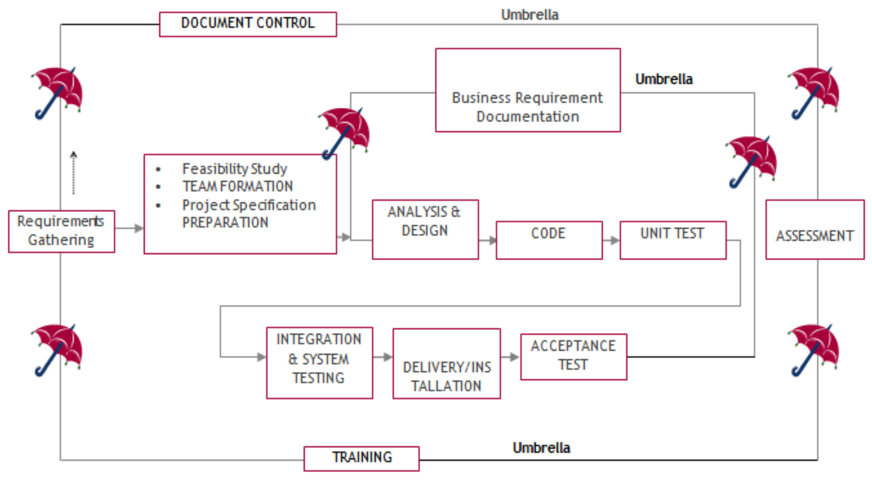


Fig.3.6: Umbrella Model

Fig. 3.6 represents SDLC, Software Development Life Cycle, it is a standardwhich is used by software industry to develop good software. Stages in SDLC include,

- Requirement Gathering
- Analysis
- Designing
- Coding
- Testing
- Maintenance

3.4.1 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.

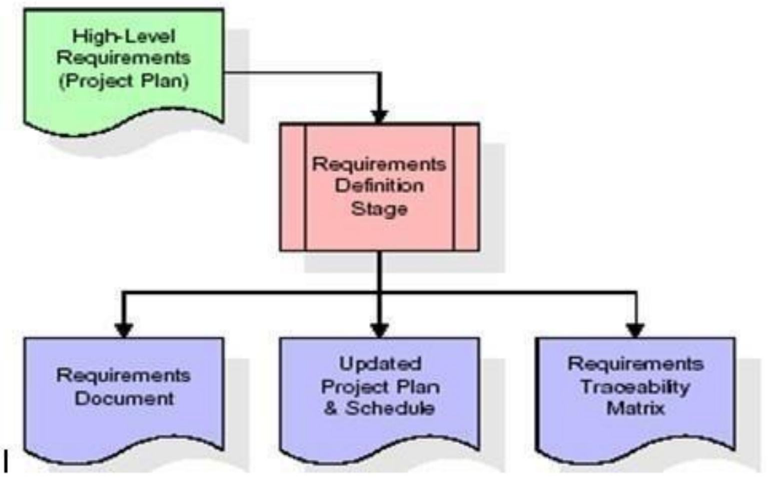

High-Level Requirements (Project Plan)
Requirements Definition Stage
Requirements Document
Updated Project Plan & Schedule
Requirements Traceability Matrix

Fig.3.7: Requirements for gathering stage

Fig. 3.7 represents requirements, and 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 not included 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.

3.4.2 ANALYSIS STAGE

The planning stage as shown in the Fig. 3.8 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.

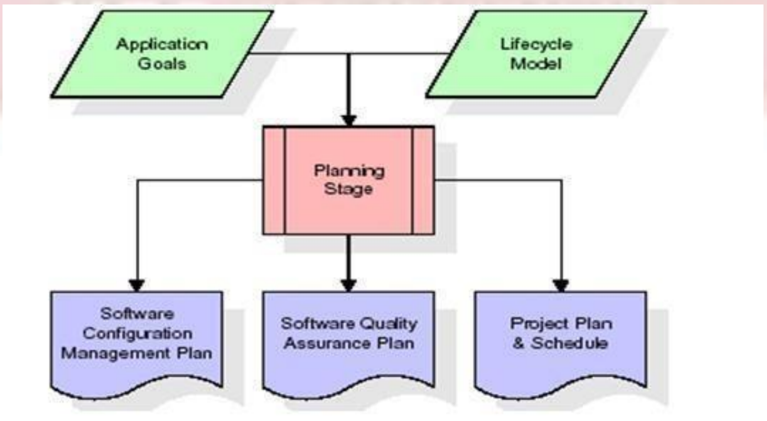

Application Goals
Lifecycle Model
Planning Stage
Software Configuration Management Plan
Software Quality Assurance Plan
Project Plan & Schedule

Fig.3.8: Analysis Stage

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.

3.4.3 DESIGNING STAGE

The design stage shown in Fig. 3.9 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 because 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.

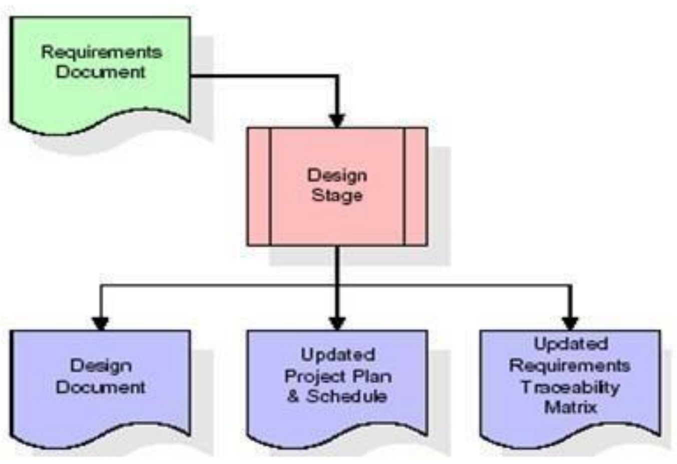

Requirements Document
Design Stage
Design Document
Updated Project Plan & Schedule
Updated Requirements Traceabilty Matrix

Fig.3.9: Designing stage

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.

3.4.4 DEVELOPMENT (CODING) STAGE

The Fig. 3.10 depicts 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.

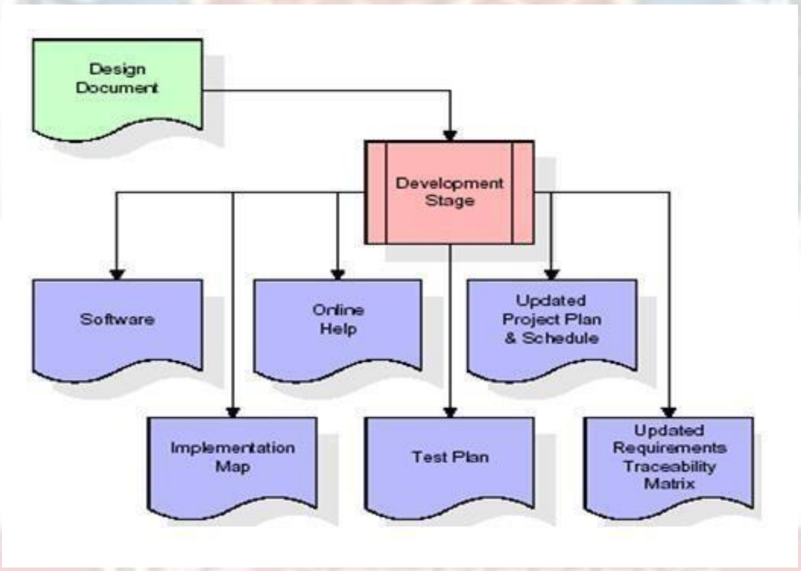


Fig.3.10: Development Stage

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.

3.4.5 INTEGRATION & TEST STAGE

During the integration and test stage shown in the Fig. 3.11, 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.

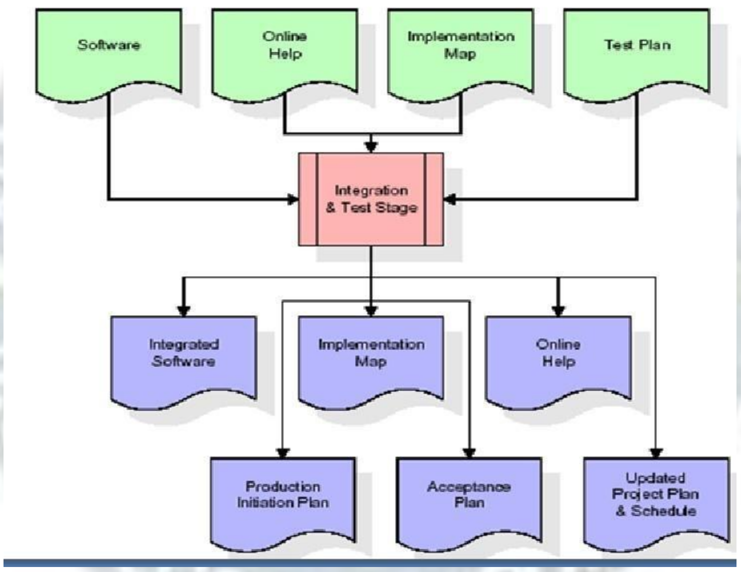


Fig.3.11: Integration and test Stage

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.

3.4.6 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.

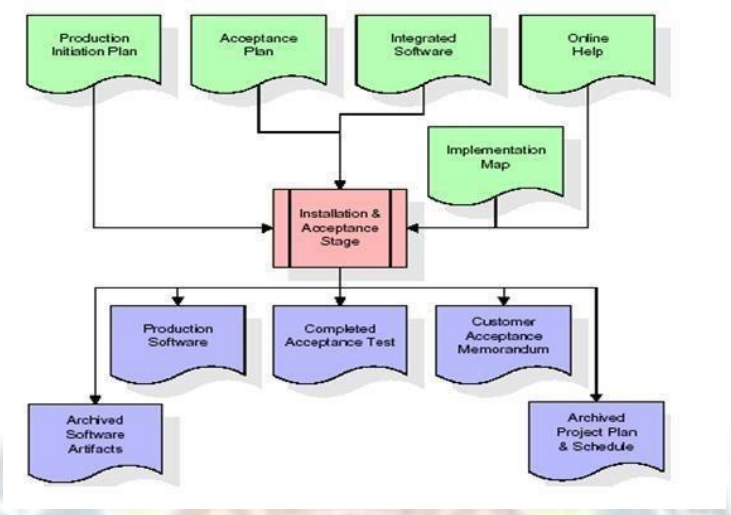

Production Initiation Plan
Acceptance Plan
Integrated Software
Online Help
Implementation Map
Installation & Acceptance Stage
Production Software
Completed Acceptance Test
Customer Acceptance Memorandum
Archived Software Artifacts
Archived Project Plan & Schedule

Fig. 3.12: Installation and Acceptance Test

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 as shown in the Fig. 3.12. Finally, the PDR enters the last of the actual labor data into the project schedule and locks the project as a permanent project record.

3.4.7 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).

3.5 FEASIBILITY STUDY

Feasibility studies aim to uncover the strengths and weaknesses of the existing system or proposed venture objectively and rationally. In its simplest term, the two criteria to judge feasibility are cost required and value to be attained. As such, a well-designed feasibility study should provide historical background of the project.

Generally, feasibility studies precede technical development and project implementation. The assessment of feasibility study is based on the following factors:

1. Technical Feasibility
2. Operational Feasibility

3.5.1 TECHNICAL FEASIBILTY

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.

3.5.2 OPERATIONAL FEASIBILTY

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.

4. IMPLEMENTATION

4.1 INTRODUCTION TO 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.

4.2 HISTORY OF PYTHON

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

4.3 FEATURES OF PYTHON

A simple language which is easier to learn Python has a quite simple and elegant syntax. It is 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 an open source You can freely use and distribute Python, even for commercial use. Not only can you use and distribute software written in it, but you can also 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 do not 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 do not need to worry about any lower-level operations. Large standard libraries to solve common tasks Python has several 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 MySQL dB library using import MySQL db. Standard libraries in Python are well tested and used by hundreds of people. So, you can be sure that it will not break your application. Object- oriented Everything in Python is an object. Object oriented programming (OOP) helpsyou solve a complex problem intuitively. With OOP, you can divide these complex problems into smaller sets by creating objects.

4.4 APPLICATIONS OF PYTHON

- It is 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)
```

- You do not 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.
- Python allows you to write programs having greater functionality with fewer lines of code.

4.5 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

4.5.1 UPLOAD ANDROID DATASET

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

4.5.2 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.

4.5.3 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.

4.5.4 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.

4.5.5 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.

4.5.6 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.

4.6 SOURCE 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()
```

4.7 SCREENSHOTS

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

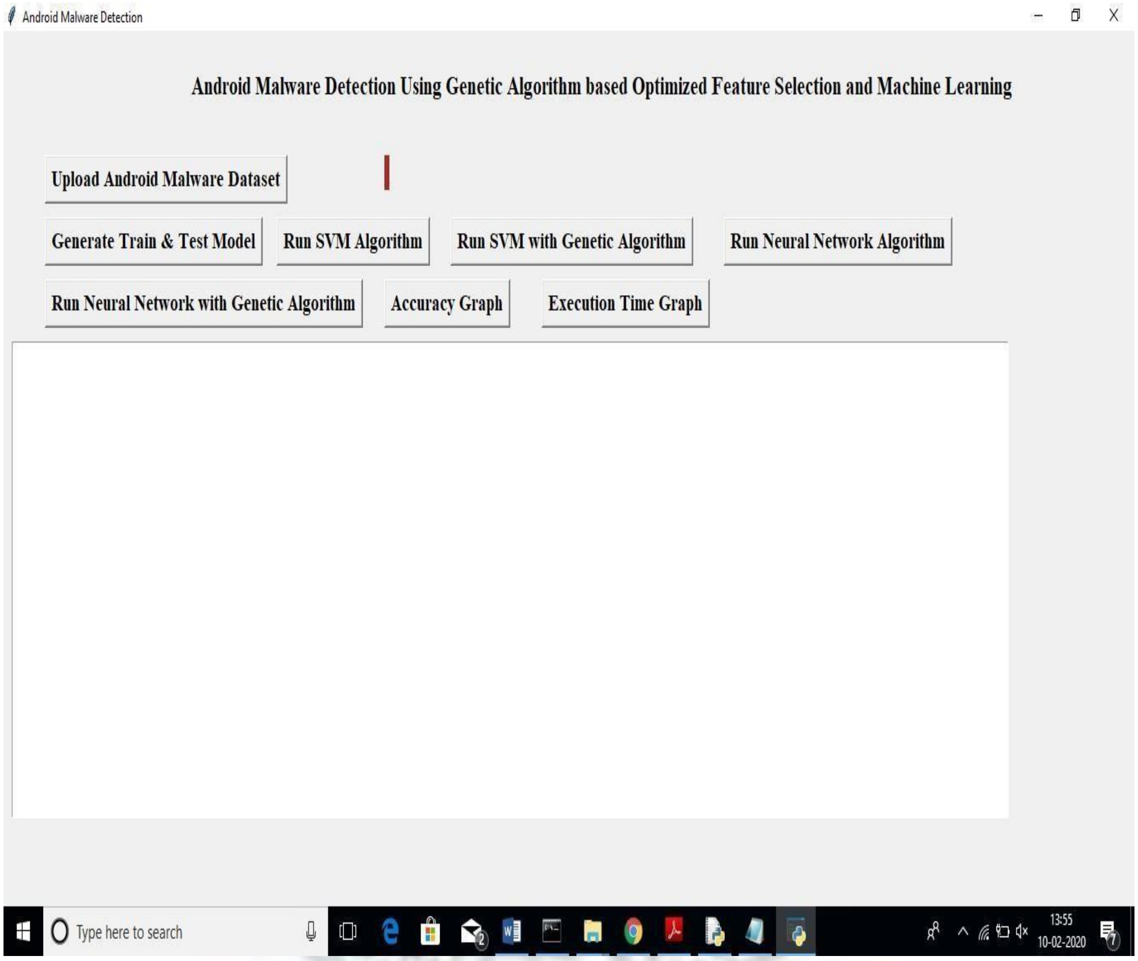

Android Malware Detection
Android Malware Detection Using Genetic Algorithm based Optimized Feature Selection and Machine Learning
Upload Android Malware Dataset
Generate Train & Test Model
Run SVM Algorithm
Run SVM with Genetic Algorithm
Run Neural Network Algorithm
Run Neural Network with Genetic Algorithm
Accuracy Graph
Execution Time Graph
Type here to search
13:55
10-02-2020

Fig. 4.1: Run Batch File (run.bat)

In above Fig. 4.1, (screen) click on 'Upload Android Malware Dataset' button and upload dataset. As soon as the batch file is clicked it opens a window, which contains all the buttons, which are defined in the program. There are 6 buttons, and a console is also opened at the behind, all the backend process is taken place in the console.

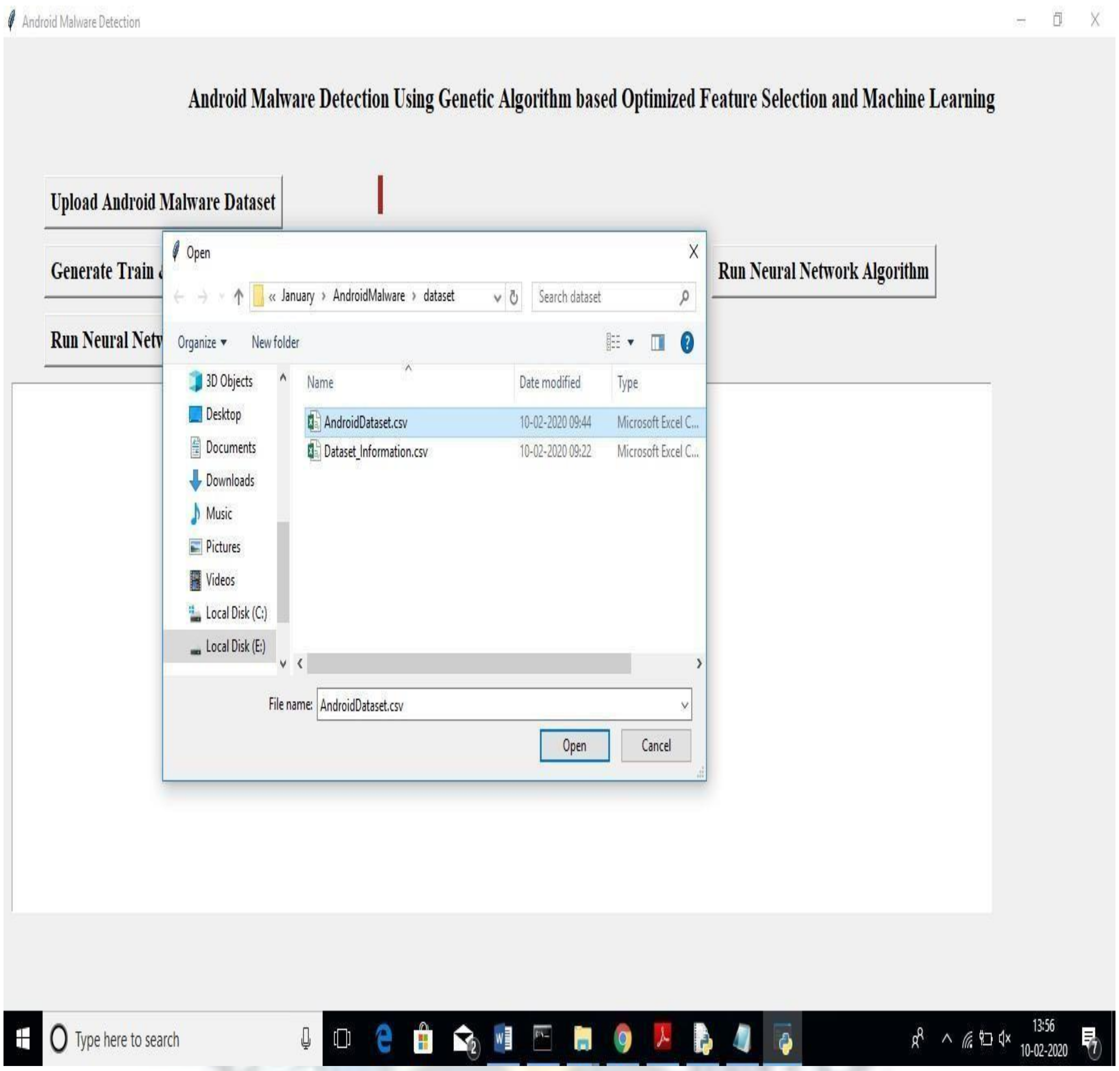
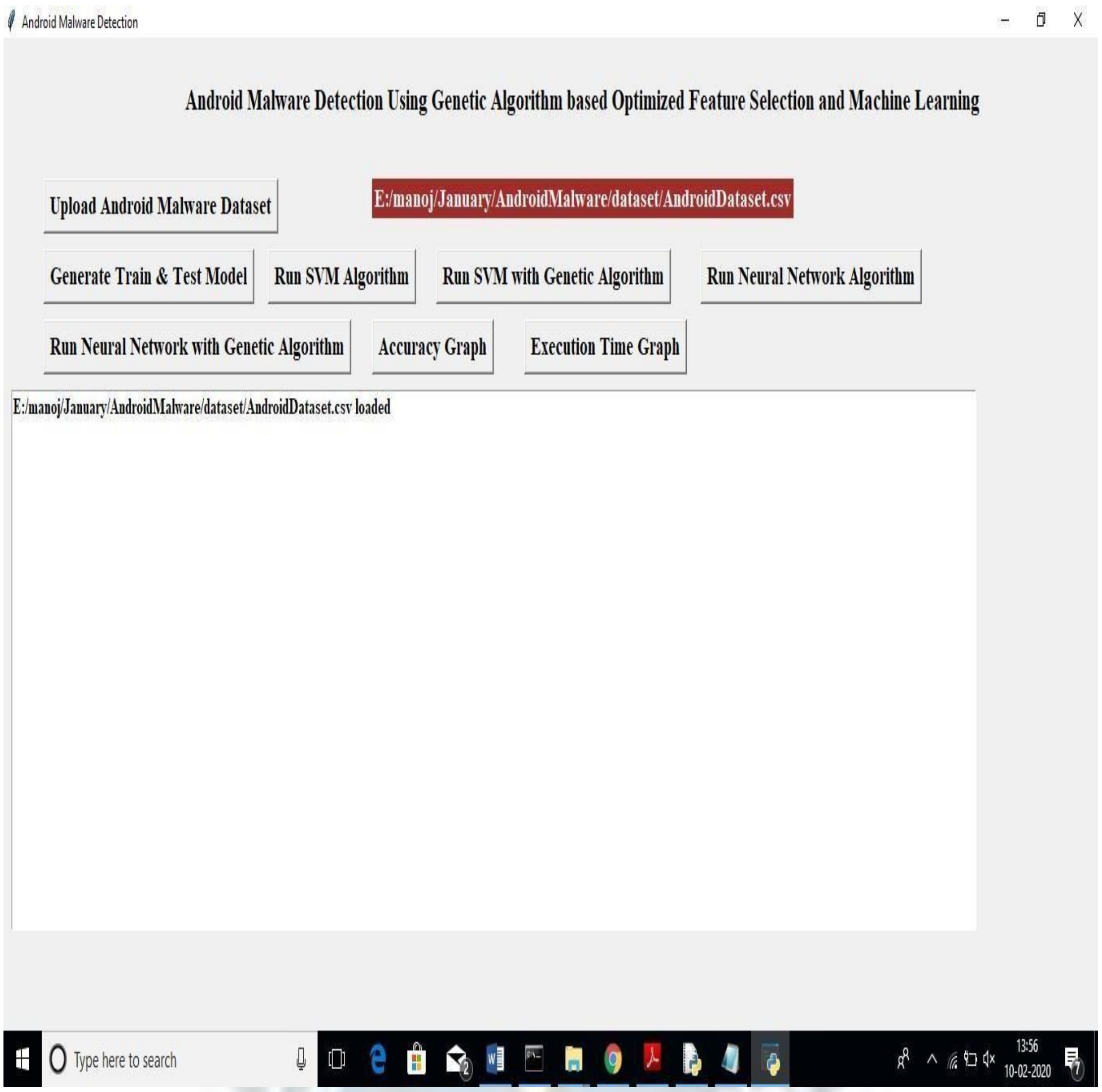


Fig. 4.2: Upload Data Set

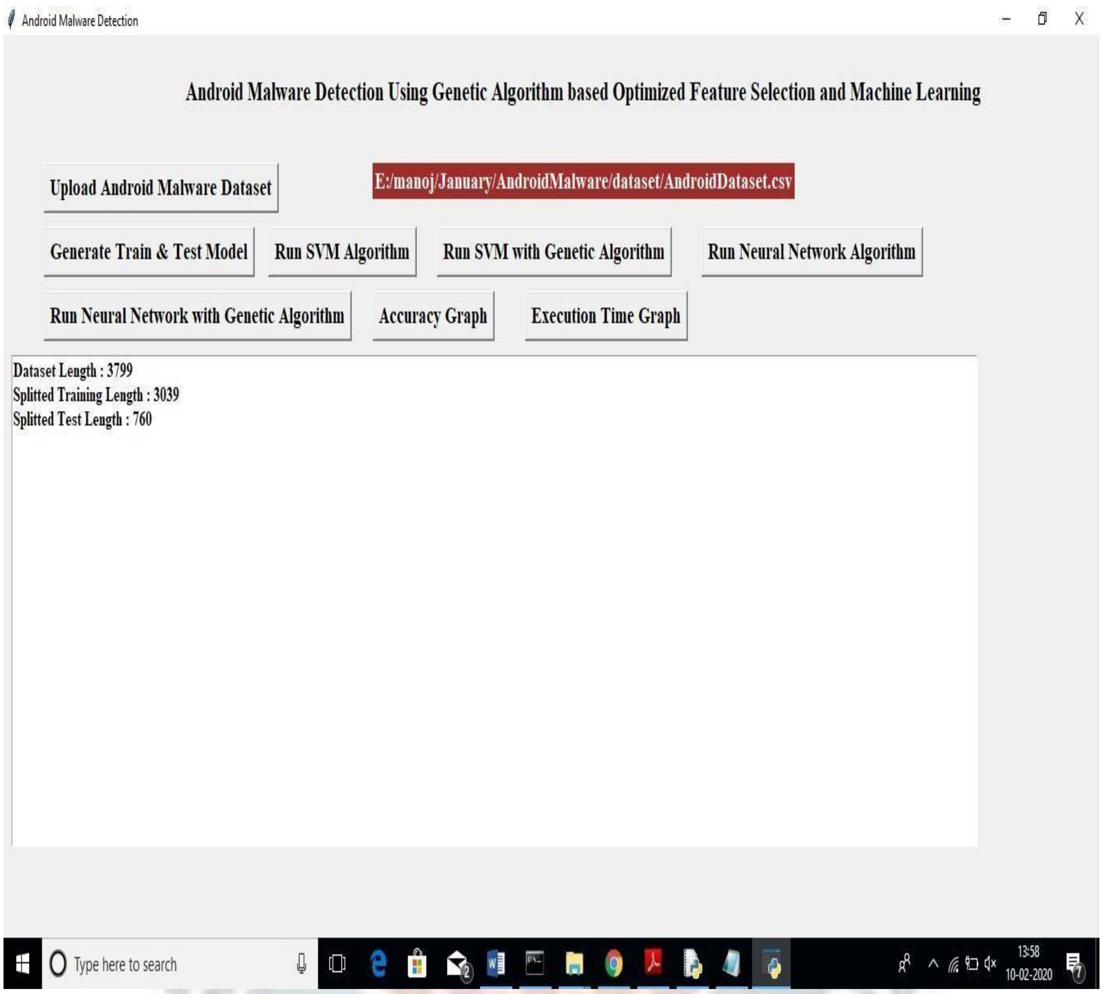
In above Fig. 4.2 the user uploads ‘AndroidDataset.csv’ file and after upload will get below screen. The file “Android Malware Dataset” contains all the features of good ware and malware which are extracted by the various tools. This file is fed as an input to the machine learning model. The dataset is downloaded from the “kaggle” website which has many datasets. The “Android Dataset” is in csv format which has features in the form of zeros and ones, there are nearly 3799 apps’ features which are present, and there are 2 classes, class 0 represents malware and 1 represents good ware.


Android Malware Detection
Android Malware Detection Using Genetic Algorithm based Optimized Feature Selection and Machine Learning
Upload Android Malware Dataset
E:/manoj/January/AndroidMalware/dataset/AndroidDataset.csv
Generate Train & Test Model
Run SVM Algorithm
Run SVM with Genetic Algorithm
Run Neural Network Algorithm
Run Neural Network with Genetic Algorithm
Accuracy Graph
Execution Time Graph
Type here to search
13:56
10-02-2020

```
E:/manoj/January/AndroidMalware/dataset/AndroidDataset.csv loaded
```

Fig. 4.3: Generate Train & Test Model

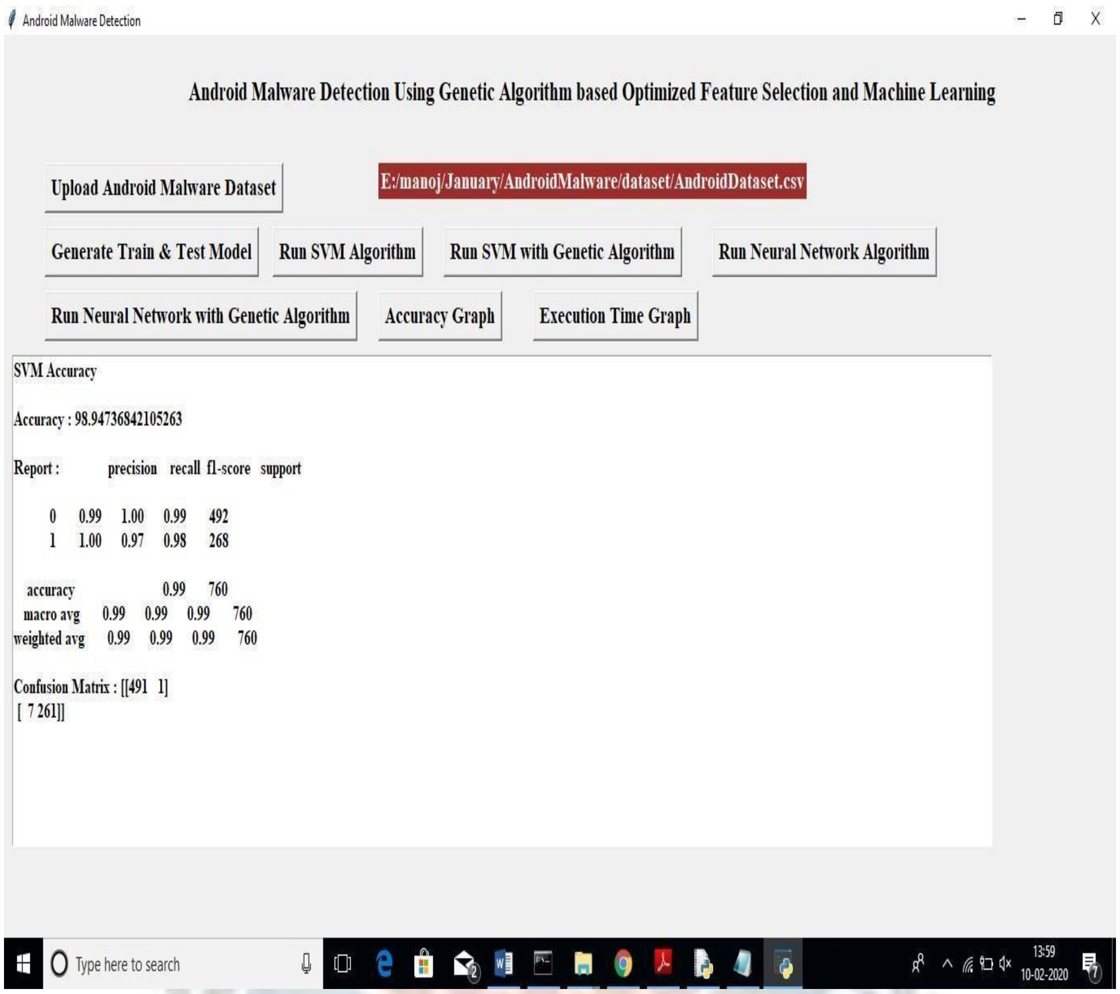
In the Fig. 4.3, when the user clicks on ‘Generate Train & Test Model’ button to split dataset into train and test part. All machine learning algorithms will take 80% dataset for training and 20% dataset to test accuracy of trained model. Model training is an important phase in machine learning.


Android Malware Detection
Android Malware Detection Using Genetic Algorithm based Optimized Feature Selection and Machine Learning
Upload Android Malware Dataset
E:/manoj/January/AndroidMalware/dataset/AndroidDataset.csv
Generate Train & Test Model
Run SVM Algorithm
Run SVM with Genetic Algorithm
Run Neural Network Algorithm
Run Neural Network with Genetic Algorithm
Accuracy Graph
Execution Time Graph
Type here to search
13:58
10-02-2020

```
Dataset Length : 3799
Splitted Training Length : 3039
Splitted Test Length : 760
```

Fig. 4.4: Data Set Partition

In above Fig. 4.4, we can see there are total 3799 android app records are there and application using 3039 records for training and 760 records for testing.

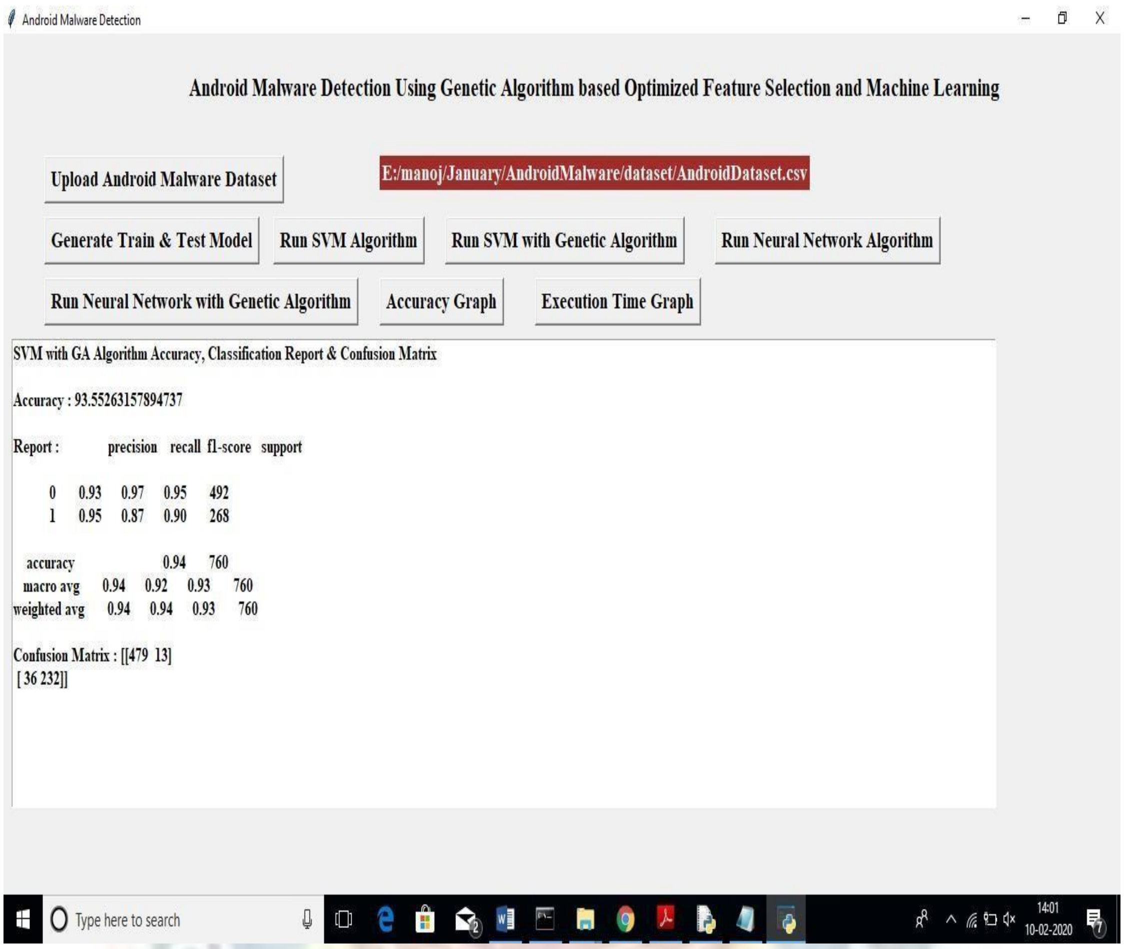

Android Malware Detection
Android Malware Detection Using Genetic Algorithm based Optimized Feature Selection and Machine Learning
Upload Android Malware Dataset
E:/manoj/January/AndroidMalware/dataset/AndroidDataset.csv
Generate Train & Test Model
Run SVM Algorithm
Run SVM with Genetic Algorithm
Run Neural Network Algorithm
Run Neural Network with Genetic Algorithm
Accuracy Graph
Execution Time Graph
Type here to search
13:59
10-02-2020

```
SVM Accuracy

Accuracy : 98.94736842105263

Report :          precision    recall  f1-score   support

           0       0.99      1.00      0.99       492
           1       1.00      0.97      0.98       268

    accuracy                           0.99       760
   macro avg       0.99      0.99      0.99       760
weighted avg       0.99      0.99      0.99       760

Confusion Matrix : [[491   1]
 [  7 261]]
```

Fig. 4.5: Run SVM Algorithm

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. SVM algorithm gives the best accuracy compared all four algorithms, but the time complexity of SVM algorithm is also high.

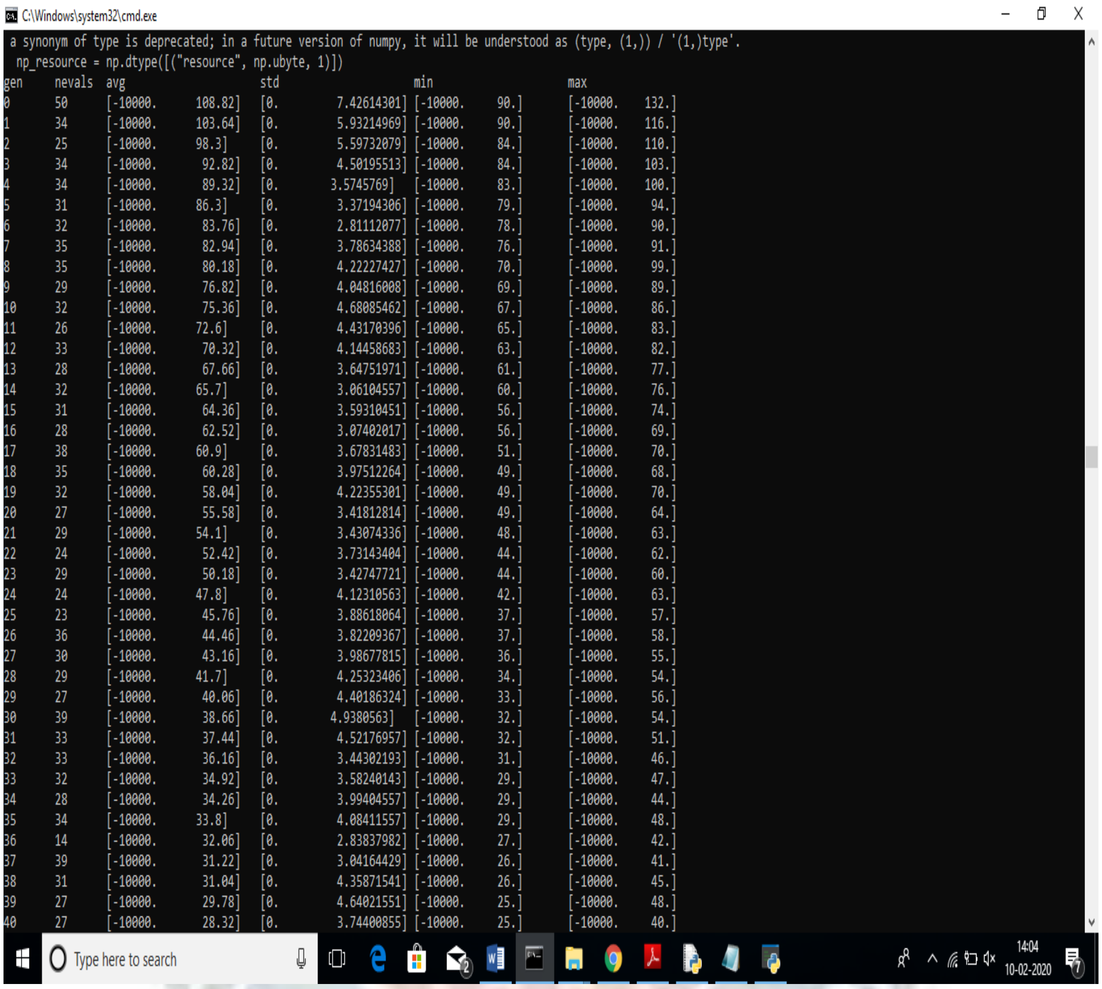
Through Fig. 4.5 uses SVM through which we got 98% accuracy for SVM and now click on ‘Run SVM with Genetic Algorithm’ button to choose optimize features and then run SVM on optimize features to get accuracy.


Android Malware Detection
Android Malware Detection Using Genetic Algorithm based Optimized Feature Selection and Machine Learning
Upload Android Malware Dataset
E:/manoj/January/AndroidMalware/dataset/AndroidDataset.csv
Generate Train & Test Model
Run SVM Algorithm
Run SVM with Genetic Algorithm
Run Neural Network Algorithm
Run Neural Network with Genetic Algorithm
Accuracy Graph
Execution Time Graph
Type here to search
14:01
10-02-2020

```
SVM with GA Algorithm Accuracy, Classification Report & Confusion Matrix

Accuracy : 93.55263157894737

Report :         precision   recall  f1-score   support

       0     0.93     0.97     0.95     492
       1     0.95     0.87     0.90     268

   accuracy                    0.94     760
  macro avg     0.94     0.92     0.93     760
weighted avg     0.94     0.94     0.93     760

Confusion Matrix : [[479  13]
 [ 36 232]]
```

Fig. 4.6: Run SVM with Genetic Algorithm

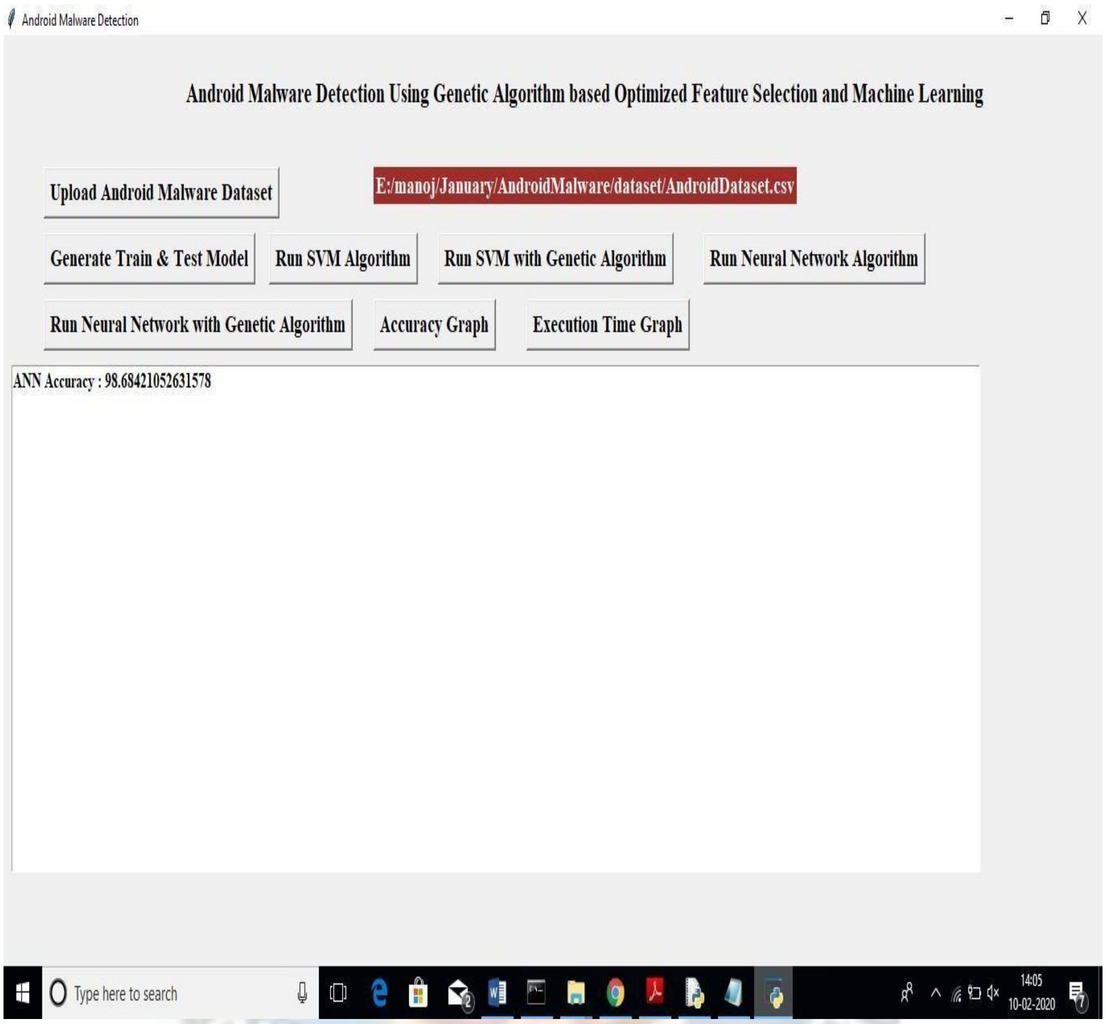
In Fig. 4.6 has SVM with Genetic algorithm got 93% accuracy. Genetic with SVM accuracy is less but its execution time will be less which we can see at the time of comparison graph.



```
C:\Windows\system32\cmd.exe
 a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
gen     nevals  avg                     std                     min                     max
0       50      [-10000.      108.82]   [0.         7.42614301] [-10000.     90.]       [-10000.    132.]
1       34      [-10000.      103.64]   [0.         5.93214969] [-10000.     90.]       [-10000.    116.]
2       25      [-10000.      98.3]     [0.         5.59732079] [-10000.     84.]       [-10000.    110.]
3       34      [-10000.       92.82]   [0.         4.50195513] [-10000.     84.]       [-10000.    103.]
4       34      [-10000.       89.32]   [0.        3.5745769]   [-10000.     83.]       [-10000.    100.]
5       31      [-10000.      86.3]     [0.         3.37194306] [-10000.     79.]       [-10000.     94.]
6       32      [-10000.       83.76]   [0.         2.81112077] [-10000.     78.]       [-10000.     90.]
7       35      [-10000.       82.94]   [0.         3.78634388] [-10000.     76.]       [-10000.     91.]
8       35      [-10000.       80.18]   [0.         4.22227427] [-10000.     70.]       [-10000.     99.]
9       29      [-10000.       76.82]   [0.         4.04816008] [-10000.     69.]       [-10000.     89.]
10      32      [-10000.       75.36]   [0.         4.68085462] [-10000.     67.]       [-10000.     86.]
11      26      [-10000.      72.6]     [0.         4.43170396] [-10000.     65.]       [-10000.     83.]
12      33      [-10000.       70.32]   [0.         4.14458683] [-10000.     63.]       [-10000.     82.]
13      28      [-10000.       67.66]   [0.         3.64751971] [-10000.     61.]       [-10000.     77.]
14      32      [-10000.      65.7]     [0.         3.06104557] [-10000.     60.]       [-10000.     76.]
15      31      [-10000.       64.36]   [0.         3.59310451] [-10000.     56.]       [-10000.     74.]
16      28      [-10000.       62.52]   [0.         3.07402017] [-10000.     56.]       [-10000.     69.]
17      38      [-10000.      60.9]     [0.         3.67831483] [-10000.     51.]       [-10000.     70.]
18      35      [-10000.       60.28]   [0.         3.97512264] [-10000.     49.]       [-10000.     68.]
19      32      [-10000.       58.04]   [0.         4.22355301] [-10000.     49.]       [-10000.     70.]
20      27      [-10000.       55.58]   [0.         3.41812814] [-10000.     49.]       [-10000.     64.]
21      29      [-10000.      54.1]     [0.         3.43074336] [-10000.     48.]       [-10000.     63.]
22      24      [-10000.       52.42]   [0.         3.73143404] [-10000.     44.]       [-10000.     62.]
23      29      [-10000.       50.18]   [0.         3.42747721] [-10000.     44.]       [-10000.     60.]
24      24      [-10000.      47.8]     [0.         4.12310563] [-10000.     42.]       [-10000.     63.]
25      23      [-10000.       45.76]   [0.         3.88618064] [-10000.     37.]       [-10000.     57.]
26      36      [-10000.       44.46]   [0.         3.82209367] [-10000.     37.]       [-10000.     58.]
27      30      [-10000.       43.16]   [0.         3.98677815] [-10000.     36.]       [-10000.     55.]
28      29      [-10000.      41.7]     [0.         4.25323406] [-10000.     34.]       [-10000.     54.]
29      27      [-10000.       40.06]   [0.         4.40186324] [-10000.     33.]       [-10000.     56.]
30      39      [-10000.       38.66]   [0.        4.9380563]   [-10000.     32.]       [-10000.     54.]
31      33      [-10000.       37.44]   [0.         4.52176957] [-10000.     32.]       [-10000.     51.]
32      33      [-10000.       36.16]   [0.         3.44302193] [-10000.     31.]       [-10000.     46.]
33      32      [-10000.       34.92]   [0.         3.58240143] [-10000.     29.]       [-10000.     47.]
34      28      [-10000.       34.26]   [0.         3.99404557] [-10000.     29.]       [-10000.     44.]
35      34      [-10000.      33.8]     [0.         4.08411557] [-10000.     29.]       [-10000.     48.]
36      14      [-10000.       32.06]   [0.         2.83837982] [-10000.     27.]       [-10000.     42.]
37      39      [-10000.       31.22]   [0.         3.04164429] [-10000.     26.]       [-10000.     41.]
38      31      [-10000.       31.04]   [0.         4.35871541] [-10000.     26.]       [-10000.     45.]
39      27      [-10000.       29.78]   [0.         4.64021551] [-10000.     25.]       [-10000.     48.]
40      27      [-10000.       28.32]   [0.         3.74400855] [-10000.     25.]       [-10000.     40.]
```

Fig. 4.7: SVM with Genetic Algorithm Backend

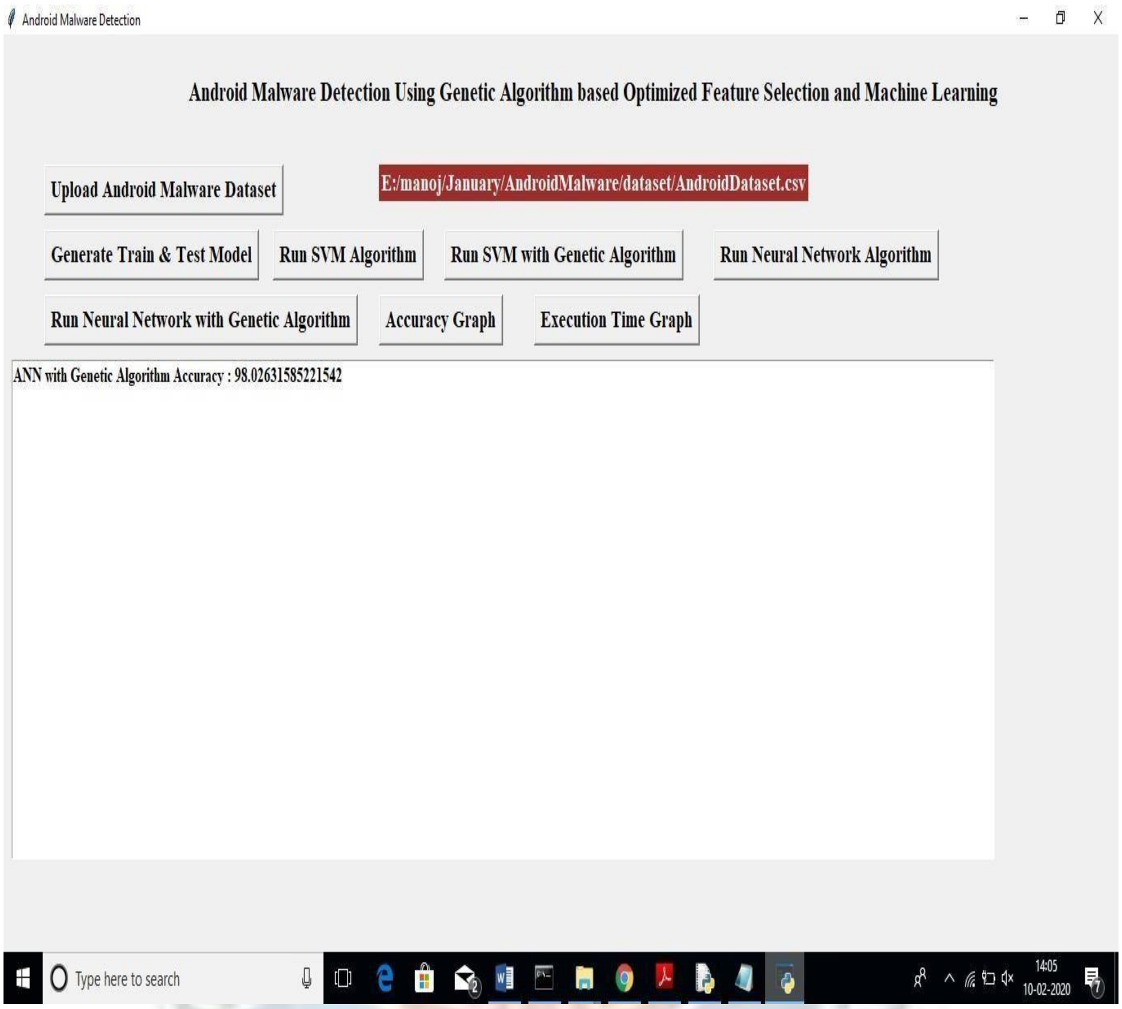
In Fig. 4.7 console we can see genetic algorithm chooses 40 features from all dataset features. Now click on ‘Run Neural Network Algorithm’ button to test neural network accuracy.


Android Malware Detection
Android Malware Detection Using Genetic Algorithm based Optimized Feature Selection and Machine Learning
Upload Android Malware Dataset
E:/manoj/January/AndroidMalware/dataset/AndroidDataset.csv
Generate Train & Test Model
Run SVM Algorithm
Run SVM with Genetic Algorithm
Run Neural Network Algorithm
Run Neural Network with Genetic Algorithm
Accuracy Graph
Execution Time Graph
Type here to search
14:05
10-02-2020

```
ANN Accuracy : 98.68421052631578
```

Fig. 4.8: Run ANN

In above Fig. 4.8 Neural Network also gave 98.64% accuracy. The Neural Network accuracy is almost equal to SVM but the time complexity taken by ANN algorithm is more when compared to all four algorithms. Now click on 'Run Neural Network with Genetic Algorithm' button to get NN accuracy with genetic algorithm.


Android Malware Detection
Android Malware Detection Using Genetic Algorithm based Optimized Feature Selection and Machine Learning
Upload Android Malware Dataset
E:/manoj/January/AndroidMalware/dataset/AndroidDataset.csv
Generate Train & Test Model
Run SVM Algorithm
Run SVM with Genetic Algorithm
Run Neural Network Algorithm
Run Neural Network with Genetic Algorithm
Accuracy Graph
Execution Time Graph
Type here to search
14:05
10-02-2020

```
ANN with Genetic Algorithm Accuracy : 98.02631585221542
```

Fig. 4.9: Run ANN with Genetic Algorithm

In above Fig. 4.9 NN with genetic got 98.02% accuracy. Now click on 'Accuracy Graph' button to see all algorithms accuracy in graph. Whenever the genetic algorithm is used, then the features are sorted out and only the discriminant features are selected from the dataset. In our program the maximum features selected are 5.

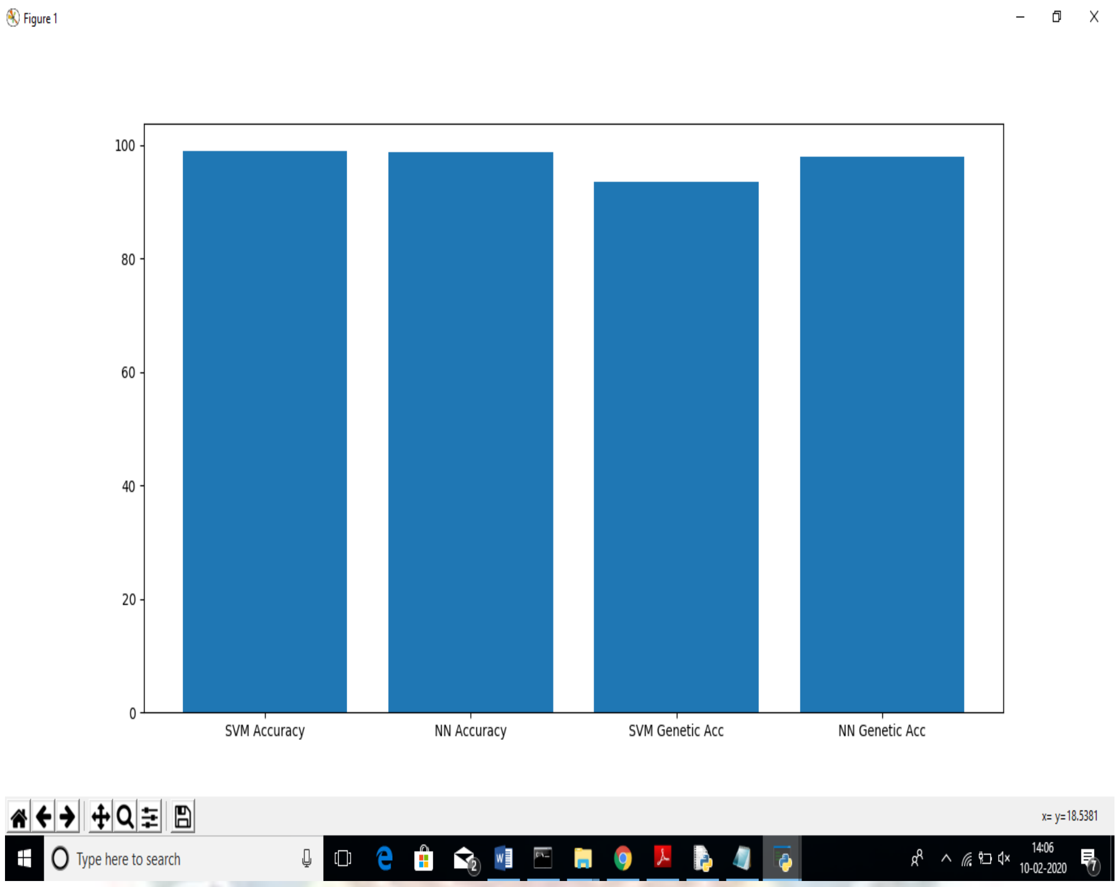


Fig. 4.10: Accuracy graph

In above Fig. 4.10, there is a graph where x-axis represents algorithm name and y-axis represents accuracy and in all SVM got high accuracy. The accuracy of all the four algorithms is almost the same, the difference lies in their time complexities. Now click on ‘Execution Time Graph’ button to get execution time of all algorithms.

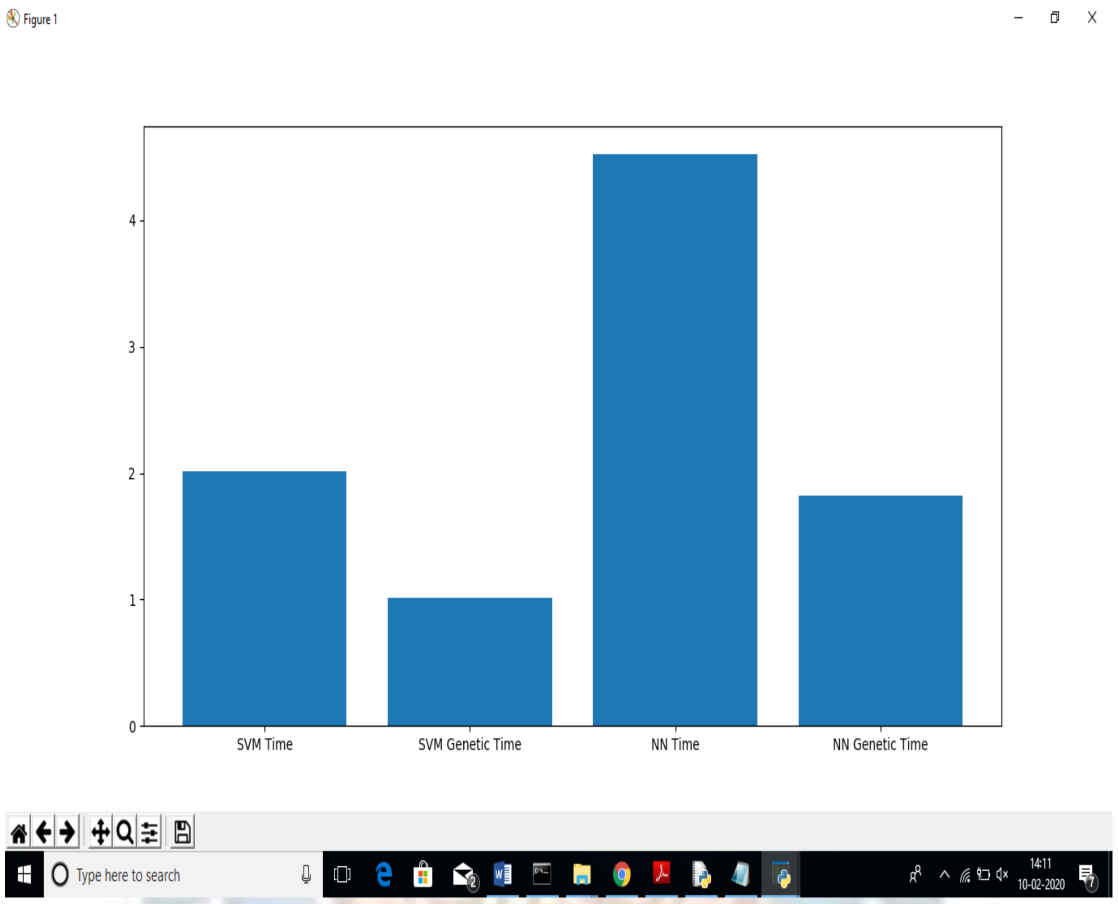


Fig. 4.11: Execution Time Graph

In above Fig. 4.11 there exists a graph in which 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. When we have combined the algorithms with genetic algorithm then the time complexity has reduced rapidly, and the accuracy did not affect much. From these 2 graphs we can conclude that genetic algorithm places an important role in machine learning models.

5. PROJECT TESTING

5.1 IMPLEMENTATION AND TESTING

Implementation is one of the most important tasks in project is the phase in which one must 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.

5.2 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 modifying 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.

5.3 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 function as a unit. The test data should be chosen such that it passed through all possible condition. 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.

5.4 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 withstand 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.

5.5 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.

5.6 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.

5.7 ACCEPTANCE TESTING

Table 5.1 Acceptance Test

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	If the Android Malware Dataset may not upload	we cannot do any further operations	we can do further operation	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 berun.	we cannot do any further operations	we can runSVM 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 algorithm.	We cannot get Accuracy graph	we can get Accuracy graph	High	High

Test Case Id	Test Case Name	Test Case Desc.	Test Steps			Test Case	Test Priority
			Step	Expected	Actual		
08	Execution Time Graph	verify the Execution Time Graph displayed or not	Without saving the Graph values of each algorithm.	We cannot get Execution Time graph	we can get Execution Time graph	High	High

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.

6. RESULTS

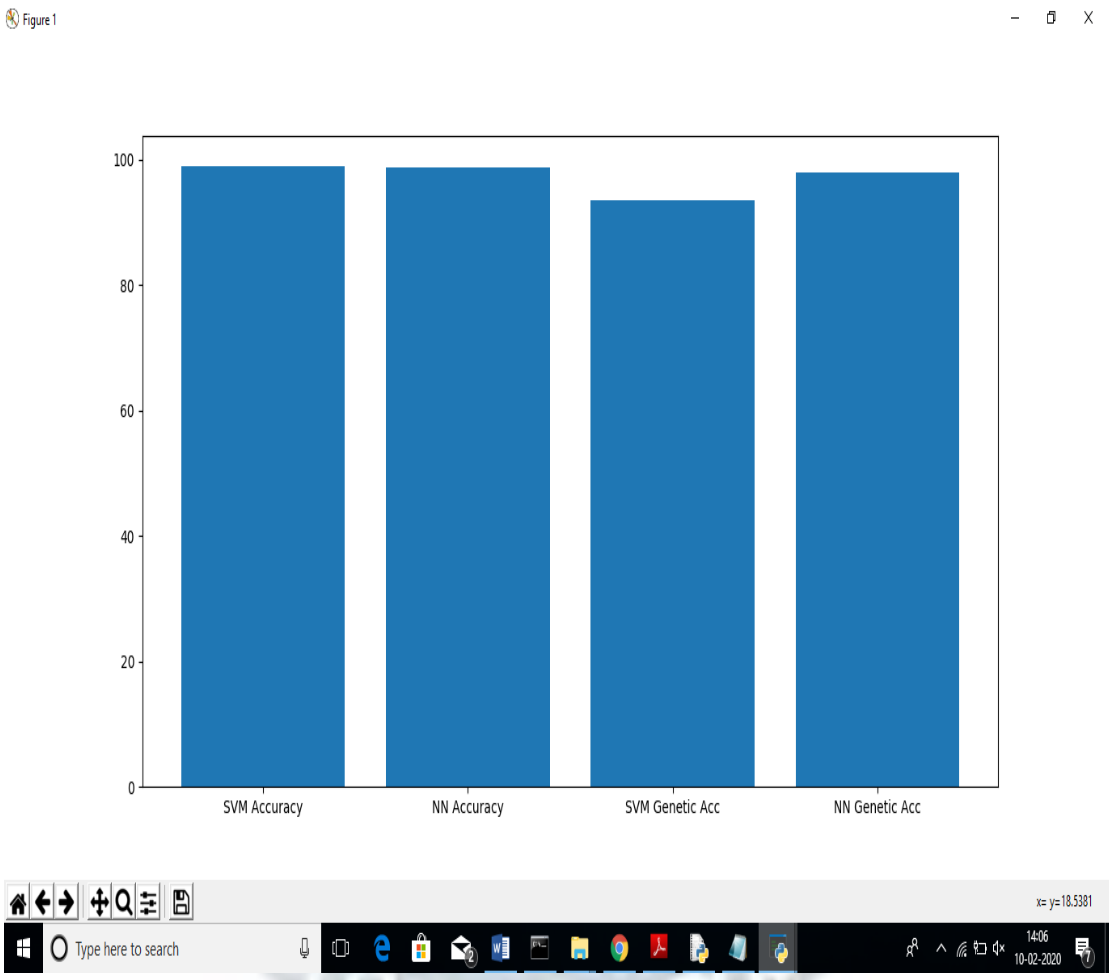


Fig. 6.1: Execution Time Graph

In Fig. 6.1 the graph has 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 algorithms.

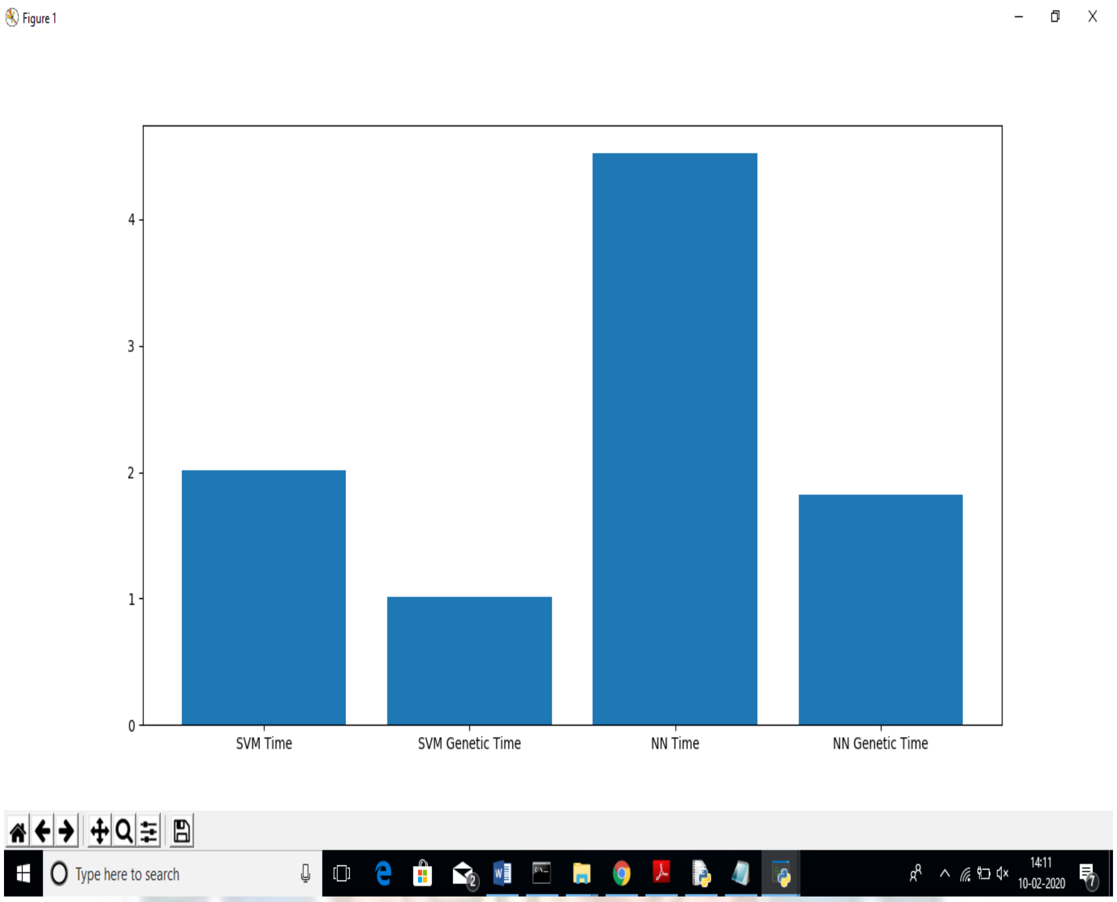


Fig. 6.2: Accuracy Graph

In above Fig. 6.2 is a graph that has 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.

7. CONCLUSION & FUTURE SCOPE

7.1 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 especially 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, 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.

7.2 FUTURE SCOPE

In future enhancements we would like to make the software run at runtime and built strategies that would have capabilities to detect new malware which attack the application and pry on user’s sensitive information.

8. REFERENCES

[1] 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.

[2] Saracino, D. Sgandurra, G. Dini, and F. Martinelli, ―MADAM: Effective and Efficient Behaviorbased Android Malware Detection and Prevention,‖ IEEE Trans. Dependable Secur. Comput., vol. 15, no. 1, pp. 83–97, 2018.

[3] TaeGuen Kim, BooJoong Kang,Mina Rho,Sakir Sezer ,Eul Gyu Im―A Multimodal Deep Learning Method for Android Malware Detection using Various Features 21 August 2018.

[4] 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.

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

[6] Alejandro Martín García, David Camacho,Valery Naranjo―Evolving Deep Neural Networks architectures for Android malware classification June 2017.

[7] X. Wang, Y. Yang, Y. Zeng, C. Tang, J. Shi, and K. Xu, "A Novel Hybrid Mobile Malware Detection System Integrating Anomaly Detection With Misuse Detection," Proc. 6th Int. Work. Mob. Cloud Comput. Serv., pp. 15–22, 2016.

[8] Y. Liu, Y. Zhang, H. Li, and X. Chen, "A hybrid malware detecting scheme for mobile Android applications," in 2016 IEEE International Conference on Consumer Electronics (ICCE), 2016,

[9] J. W. Jang, H. Kang, J. Woo, A. Mohaisen, and H. K. Kim, "AndroDumpsys: Anti-malware system based on the similarity of malware creator and malware centric information," Comput. Secur., vol. 58, pp. 125–138, 2016.

[10] Z. Yuan, Y. Lu, and Y. Xue, "Droiddetector: android malware characterization and detection using deep learning," Tsinghua Sci. Technol., vol. 21, no. 01, pp. 114–123, 2015

[11] H.-Y. Chuang and S.-D. Wang, "Machine Learning Based Hybrid Behavior Models for Android Malware Analysis," in 2015 IEEE International Conference on Software Quality, Reliability and Security, 2015.

[12] S.Y. Yerima, S. Sezer, I. Muttik, Android malware detection: An eigenspace analysis approach, in Science and Information Conference (SAI), 2015, IEEE.

[13] 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.

[14] U. Pehlivan, N. Baltaci, C. Acarturk, and N. Baykal, "The analysis of feature selection methods and classification algorithms in permission based Android malware detection," in 2014 IEEE Symposium on Computational Intelligence in Cyber Security (CICS), 2014

[15] Dafang Zhang,Wenjia Li,Kai Zhao―A Deep Learning Approach to Android Malware Feature Learning and Detection, 2014.