A

Mini Project

On

**ANDROID MALWARE DETECTION USING MACHINE LEARNING TECHNIQUES**

(Submitted in partial fulfillment of the requirements for the award of Degree)

**BACHELOR OF TECHNOLOGY**

**In**

**COMPUTER SCIENCE AND ENGINEERING**

**By**

V GOPI SINGH (217R5A0523)

THIPPAMOLA MAHESH (207R1A05P3)

PATHURI SHRUTHI REDDY (207R1A05M7)

**UNDER THE GUIDANCE OF**

## K. RANJITH REDDY

(Assistant Professor)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

## CMR TECHNICAL CAMPUS

## UGC AUTONOMOUS

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi) Recognized Under Section 2(f) & 12(B) of the UGCAct.1956, Kandlakoya (V), Medchal Road, Hyderabad-501401.

**2020-2024**

### **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



**CERTIFICATE**

This is to certify that the project entitled **“ANDROID MALWARE DETECTION USING MACHINELEARNING TECHNIQUES”** being submitted by **V GOPI SINGH (217R5A0523), THIPPAMOLA MAHESH (207R1A05P3) & PATHURI SHRUTHI REDDY (207R1A05M7)** in partialfulfillment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering to the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafide work carried out by them under our guidance and supervision during the year 2023-24.

The results embodied in this project have not been submitted to any other University or Institute for the award of any degree or diploma.

|  |  |  |
| --- | --- | --- |
| **K. Ranjith Reddy** | | **Dr. A. Raji Reddy** |
| (Assistant Professor)  INTERNAL GUIDE | | DIRECTOR |
| **Dr. K. Srujan Raju**  HOD | **EXTERNAL EXAMINER** | |

**Submitted for viva voice Examination held on \_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## ACKNOWLEDGEMENT

Apart from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We take this opportunity to express my profound gratitude and deep regard to

my guide **K.** **RANJITH REDDY,** Assistant Professor for his exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by him shall carry us a long way in the journey of life on which we are about to embark.

We also take this opportunity to express a deep sense of gratitude to the Project Review Committee (PRC) **G.Vinesh Shanker, Dr. J. Narasimharao, Ms. Shilpa, & Dr. K. Maheswari** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. K. Srujan Raju,** Head, Department of Computer Science and Engineering for providing encouragement and support for completing this project successfully. We are obliged to **Dr. A. Raji Reddy,** Director for being cooperative throughout the course of this project. We also express our sincere gratitude to Sri. **Ch. Gopal Reddy,** Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project.

The guidance and support received from all the members of **CMR Technical Campus** who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

**V GOPI SINGH (217R5A0523)**

**THIPPAMOLA MAHESH (207R1A05P3)**

**PATHURI SHRUTHI REDDY (207R1A05M7**

**ABSTRACT**

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.

**LIST OF FIGURES**

|  |  |  |  |
| --- | --- | --- | --- |
| **FIGURE NO.** | **FIGURE NAME** | **PAGE NO** | |
| Figure 3.1 | Project Architecture of Android Malware Detection Using Machine learning techniques. | 8 | |
| Figure 3.2 | Use Case Diagram for Android Malware Detection Using Machine learning techniques. | 10 | |
| Figure 3.3 | Class Diagram for Android Malware Detection Using Machine learning techniques. | 11 | |
| Figure 3.4 | Sequence diagram for Android Malware Detection Using Machine learning techniques. | 12 | |
| Figure 3.5 | Activity diagram for Android Malware Detection Using Machine learning techniques. | 13 | |
| Figure 3.6 | Component diagram for Android Malware Detection Using Machine learning techniques. | 14 | |
| Figure 3.7 | Deployment diagram for Android Malware Detection Using Machine learning techniques. | 15 | |
|  | **LIST OF SCREENSHOOTS** | |  |

|  |  |  |
| --- | --- | --- |
| **SCREENSHOT NO.** | **SCREENSHOT NAME** | **PAGE NO**. |
| Screenshot 5.1 | User Interface | 24 |
| Screenshot 5.2 | Uploading the Android dataset. | 25 |
| Screenshot 5.3 | Feature Extraction | 26 |
| Screenshot 5.4 | Train SVM Algorithm | 27 |
| Screenshot 5.5  Screenshot 5.6 | Train SVM with Genetic Algorithm  Genetic algorithm prediction of dataset | 28  29 |
| Screenshot 5.7 | Genetic algorithm chooses 40 features from all dataset features | 30 |
| Screenshot 5.8 | Train Neural Network Algorithm | 31 |
| Screenshot 5.9 | Train Neural Network with Genetic Algorithm | 32 |
| Screenshot 5.10 | Accuracy Graph | 33 |
| Screenshot 5.11 | We can save Graph for future | 34 |
| Screenshot 5.12 | Execution Time Graph | 35 |

|  |  |
| --- | --- |
| **TABLE OF CONTENTS** |  |
| **ABSTRACT** | i |
| **LIST OF FIGURES** | ii |
| **LIST OF SCREENSHOTS** | iii |
| **1. INTRODUCTION** | 1 |
| 1.1 [PROJECT SCOPE](#_bookmark0) | [1](#_bookmark0) |
| 1.2 [PROJECT PURPOSE](#_bookmark1) | [1](#_bookmark1) |
| 1.3 [PROJECT FEATURES](#_bookmark2) | [2](#_bookmark2) |
| **2.** [**SYSTEM ANALYSIS**](#_bookmark3) | [3](#_bookmark3) |
| 2.1 [PROBLEM DEFINITION](#_bookmark4) | 3 |
| 2.2 [EXISTING SYSTEM](#_bookmark5) | 3 |
| 2.2.1 DISADVANTAGES OF THE EXISTING SYSTEM | 4 |
| 2.3 [PROPOSED SYSTEM](#_bookmark6) | 4 |
| 2.3.1 ADVANTAGES OF PROPOSED SYSTEM | 4 |
| 2.4 [FEASIBILITY STUDY](#_bookmark7) | 5 |
| 2.4.1 [ECONOMIC FEASIBILITY](#_bookmark8) | 5 |
| 2.4.2 [TECHNICAL FEASIBILITY](#_bookmark9) | 5 |
| 2.4.3 SOCIAL FEASIBILITY | 6 |
| 2.5 [HARDWARE & SOFTWARE REQUIREMENTS](#_bookmark10) | 6 |
| 2.5.1 [HARDWARE REQUIREMENTS](#_bookmark11) | 6 |
| 2.5.2 SOFTWARE REQUIREMENTS | 7 |
| **3. ARCHITECTURE** | 8 |
| 3.1 PROJECT ARCHITECTURE | 8 |
| 3.2 [DESCRIPTION](#_bookmark12) | 9 |
| 3.3 [USE CASE DIAGRAM](#_bookmark13) | 10 |
| 3.4 [CLASS DIAGRAM](#_bookmark14) | 11 |
| 3.5 [SEQUENCE DIAGRAM](#_bookmark15) | 12 |
| 3.6 [ACTIVITY DIAGRAM](#_bookmark16) | 1[3](#_bookmark16) |
| 3.6 [COMPONENT DIAGRAM](#_bookmark16) | 1[4](#_bookmark16) |
| 3.6 [DEPLOYMENT DIAGRAM](#_bookmark16) | 1[5](#_bookmark16) |
| **4.** [**IMPLEMENTATION**](#_bookmark17) | [16](#_bookmark17) |
| 4.1 [SAMPLE CODE](#_bookmark18) | [16](#_bookmark18) |
| **5. SCREENSHOTS** | 24 |
| **6 .** [**TESTING**](#_bookmark19) | 36 |
| 6.1 [INTRODUCTION TO TESTING](#_bookmark20) | 36 |
| 6.2 [TYPES OF TESTING](#_bookmark21) | 36 |

* + 1. [UNIT TESTING](#_TOC_250003) 36
    2. [INTEGRATION TESTING 36](#_bookmark22)
    3. [FUNCTIONAL TESTING 37](#_bookmark23)

[6.3 TEST CASES 39](#_TOC_250002)

7. [CONCLUSION & FUTURE SCOPE](#_bookmark24) 40

* 1. [PROJECT CONCLUSION 40](#_TOC_250001)
  2. [FUTURE SCOPE](#_bookmark25) 40

8. [BIBLIOGRAPHY 41](#_TOC_250000)

|  |  |  |
| --- | --- | --- |
| 8.1 | [REFERENCES](#_bookmark26) | 41 |
| 8.2 | [GITHUB LINK](#_bookmark27) | 41 |

**1.INTRODUCTION**

1. **INTRODUCTION**

* 1. **PROJECT SCOPE**

The project scope for an “**ANDROID MALWARE DETECTION USING MACHINE LEARNING TECHNIQUES**” encompasses the development of a real-time application security solution. This system will focus on identifying and mitigating malicious Android applications by leveraging machine learning algorithms. Key components include data collection and preprocessing, machine learning model selection and training, real-time scanning, dynamic behavior analysis, and continuous learning through model updates. User interaction features, resource optimization, privacy considerations, and comprehensive documentation will also be integral parts of the project. The project's primary objective is to enhance Android device security by providing proactive malware detection, while adhering to ethical and privacy guidelines, and ensuring effective user support.

* 1. **PROJECT PURPOSE**

Android Apps are freely available on Google Play 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. Malware writers 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.

* 1. **PROJECT FEATURES**

This project encompasses a range of key features such as graph based analysis of good ware/malware, user friendly interface(user-panel). In the user-panel the user can see the malicious percentage of the application. And the processed output of the semantic analysis will be displayed to the user in the form of graph and the user will get a proper review of the application. 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.

**2. SYSTEM ANALYSIS**

1. **SYSTEM ANALYSIS**

System Analysis is the important phase in the system development process. The System is studied to the minute details and analyzed. The system analyst plays an important role of an interrogator and dwells deep into the working of the present system. In analysis, a detailed study of these operations performed by the system and their relationships within and outside the system is done. A key question considered here is, “what must be done to solve the problem?” The system is viewed as a whole and the inputs to the system are identified. Once analysis is completed the analyst has a firm understanding of what is to be done.

**2.1 PROBLEM DEFINITION**

Any installation of malware app will steal information from phone and transfer to cyber-criminals or can give total phone control to criminal’s hand. To protect users from such app we are using machine learning algorithm to detect malware from mobile app. To detect malware from app we need to extract all code from app using reverse engineering and then check whether app is doing any mischievous activity such as sending SMS or copying contact details without having proper permissions. If such activity given in code, then we will detect that app as malicious app.

**2.2 EXISTING SYSTEM**

1. **Signature-based Detection:** The signature-based detection technique uses a database of known malware signatures to detect and block malware. However, this technique is ineffective against unknown or zero-day malware.
2. **Dynamic Analysis:** The dynamic analysis technique executes an application in a controlled environment to detect malicious activities. This technique can detect known and unknown malware but requires more computational resources.

**2.2.1 DISADVANTAGES OF EXISTING SYSTEM**

* In the existing system, the application permissions are extracted to detect the malware and executed through the command prompt.
* A proper GUI was not provided to execute the tasks.
* All the commands were run through the command prompt. It was difficult for the non-technical user to use the system.
* And also, Semantic analysis was not implemented.

**2.3 PROPOSED SYSTEM**

Two set of Android Apps or APKs: Malware/Goodware are 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.

* + 1. **ADVANTAGES OF THE PROPOSED SYSTEM**
* Security
* Proposed a novel and efficient algorithm for feature selection to improve overall detection accuracy.
* 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.
  1. **FEASIBILITY STUDY**

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

Three key considerations involved in the feasibility analysis are,

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

**2.4.1 ECONOMIC FEASIBILITY**

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

**2.4.2 TECHNICAL FEASIBILITY**

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

**2.4.3 SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**2.5 HARDWARE & SOFTWARE REQUIREMENTS**

**2.5.1 HARDWARE REQUIREMENTS:**

For developing the application the following are the Hardware Requirement

* Operating System supported by
* Windows 7 or Higher
* Windows XP
* **Processo**r – Pentium IV or Intel Core i5 or i7, AMD Ryzen 5 or 7, or equivalent processors
* **RAM** -- 256 MB
* **Space on Hard Disk** -- 20 GB

**2.5.2 SOFTWARE REQUIREMENTS:**

For developing the Application the following are the software Requirement

1. **Operating System :**

* Windows 7 or Higher

**2. Programming language :**

* Python

**3. Frame Work :**

* Django

**4. Database :**

* MySQL

**5. Server :**

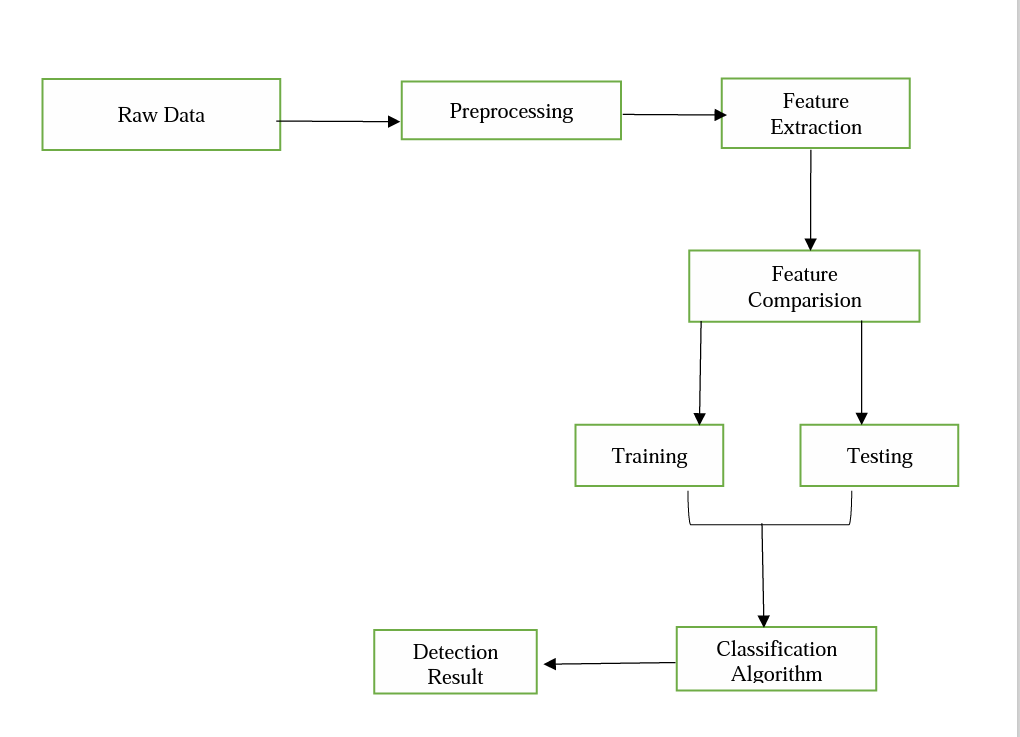
* Wamp 2

**3.ARCHITECTURE**

**3.ARCHITECTURE**

**3.1 PROJECT ARCHITECTURE**

This project architecture shows the procedure followed for classification, starting from input to final prediction.



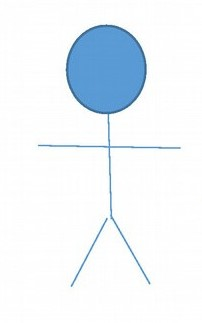
**Figure 3.1**: Architecture of Android malware detection

**3.2 DESCRIPTION**

In this project we are using two machine learning algorithms such as SVM (Support Vector Machine) and NN (Neural Networks). App will contains more than 100 features and machine learning will take more time to build model so we need to optimized (reduce dataset columns size) features, to optimized features author is using genetic algorithm. Genetic algorithm will choose important features from dataset to train model and remove un-important features. Due to this process dataset size will be reduced and training model will be generated faster. In this paper comparison we are losing some accuracy after applying genetic algorithm but we are able to reduce model training execution time.

**3.3 USE CASE DIAGRAM**

In the use case diagram, we have basically one actor who is the user in the trained model. A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures.

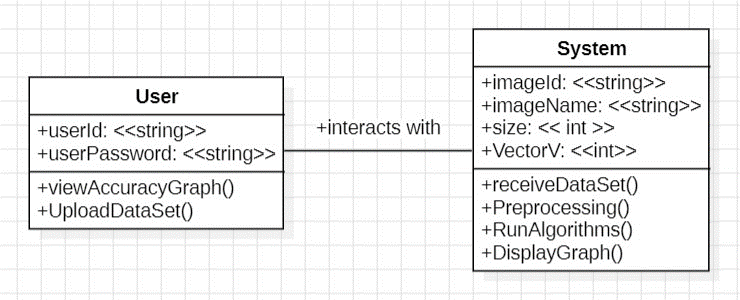


System User

**Figure 3.2:** Use Case Diagram for Android malware detection

**3.4 CLASS DIAGRAM**

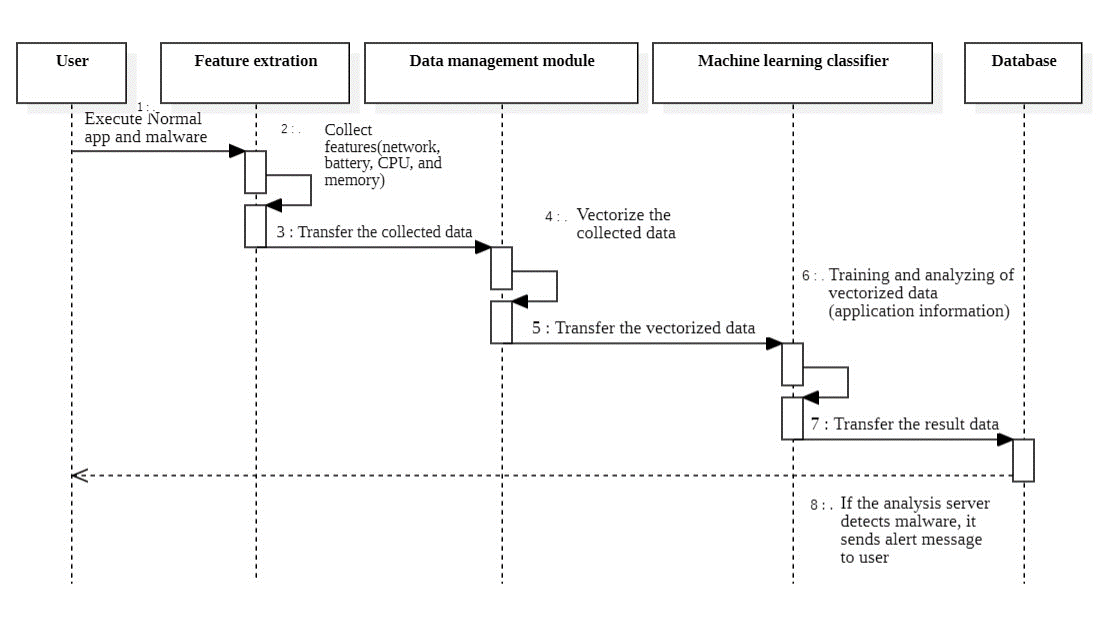
Class diagram is a type of static structure diagram that describes the structure of a system by showing the system’s classes, their attributes, operations(or methods), and the relationships among objects.



**Figure 3.3**: Class Diagram for Android malware detection

**3.5 SEQUENCE DIAGRAM**

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects 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.



**Figure 3.4**: Sequence Diagrams for Android malware detection

**3.6 ACTIVITY DIAGRAM**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. They can also include elements showing the flow of data between activities through one or more data stores.

Upload Dataset

Preprocessing

Run SVM Alg

Run SVM with Genetic Alg

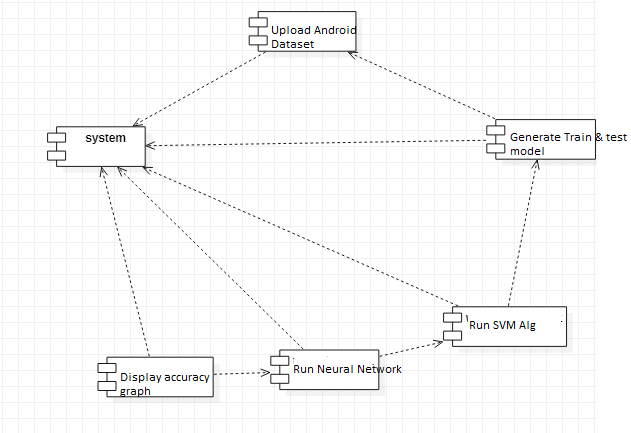
Run Neuran Network Alg

Run Neuran Network with Genetic Alg

Display Graph

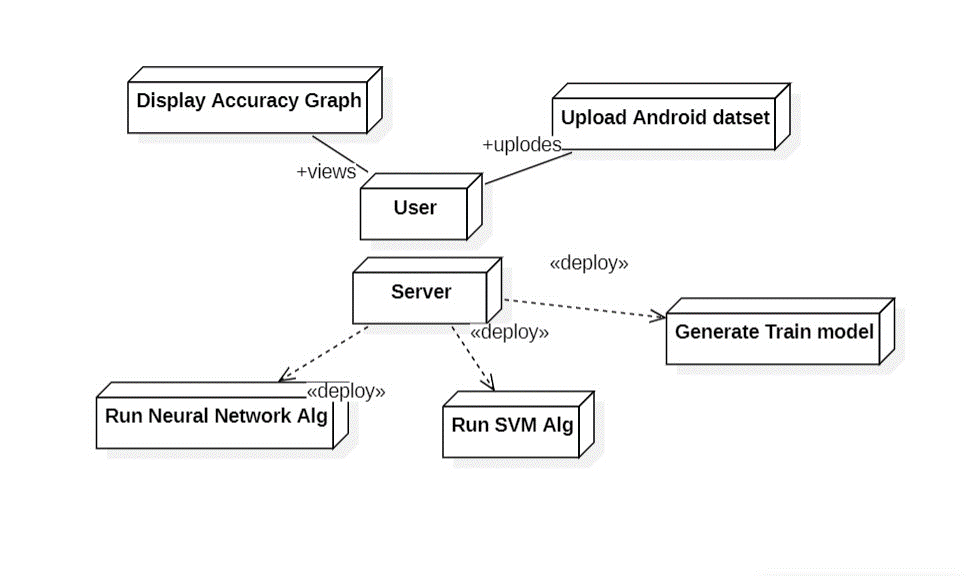
**Figure 3.5**: Activity Diagram for Android malware detection

**3.7 COMPONENT DIAGRAM**

****

**Figure 3.6**: Component Diagram for Android malware detection.

**3.8 DEPLOYMENT DIAGRAM**



**Figure 3.7**: Deployment Diagram for Android malware detection.

**4. IMPLEMENTATION**

**4.IMPLEMENTATION**

**4.1 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)

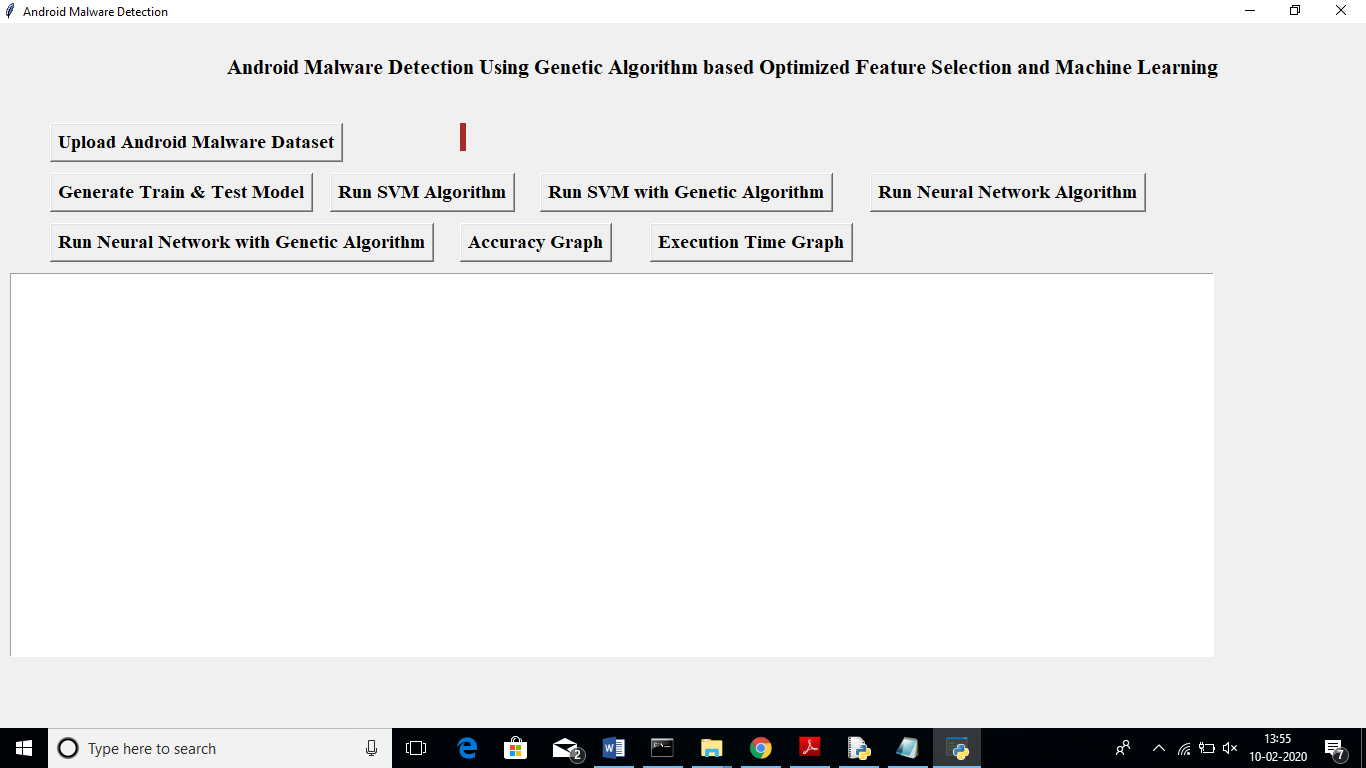
#main.config()

main.mainloop()

**5. SCREENSHOTS**

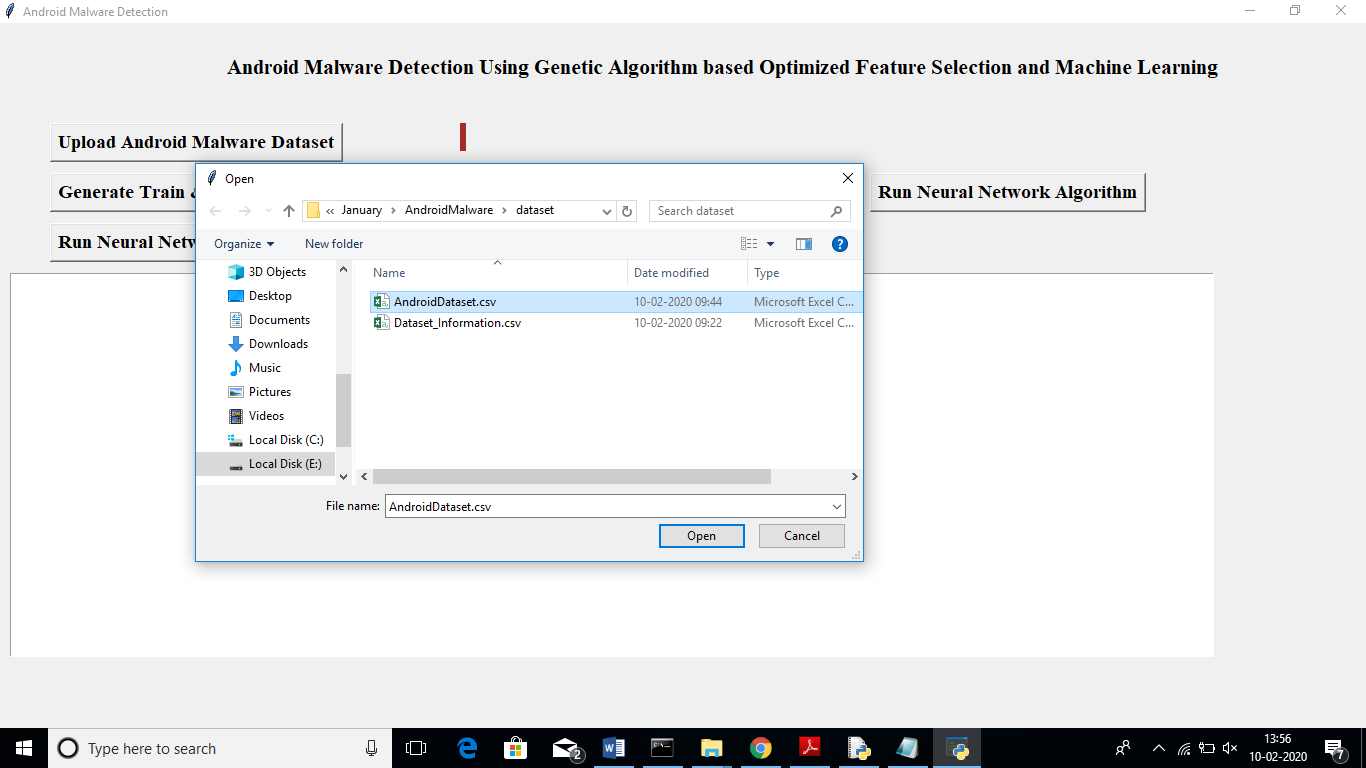
**5.SCREENSHOTS**

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



Screenshot 5.1 : User Interface

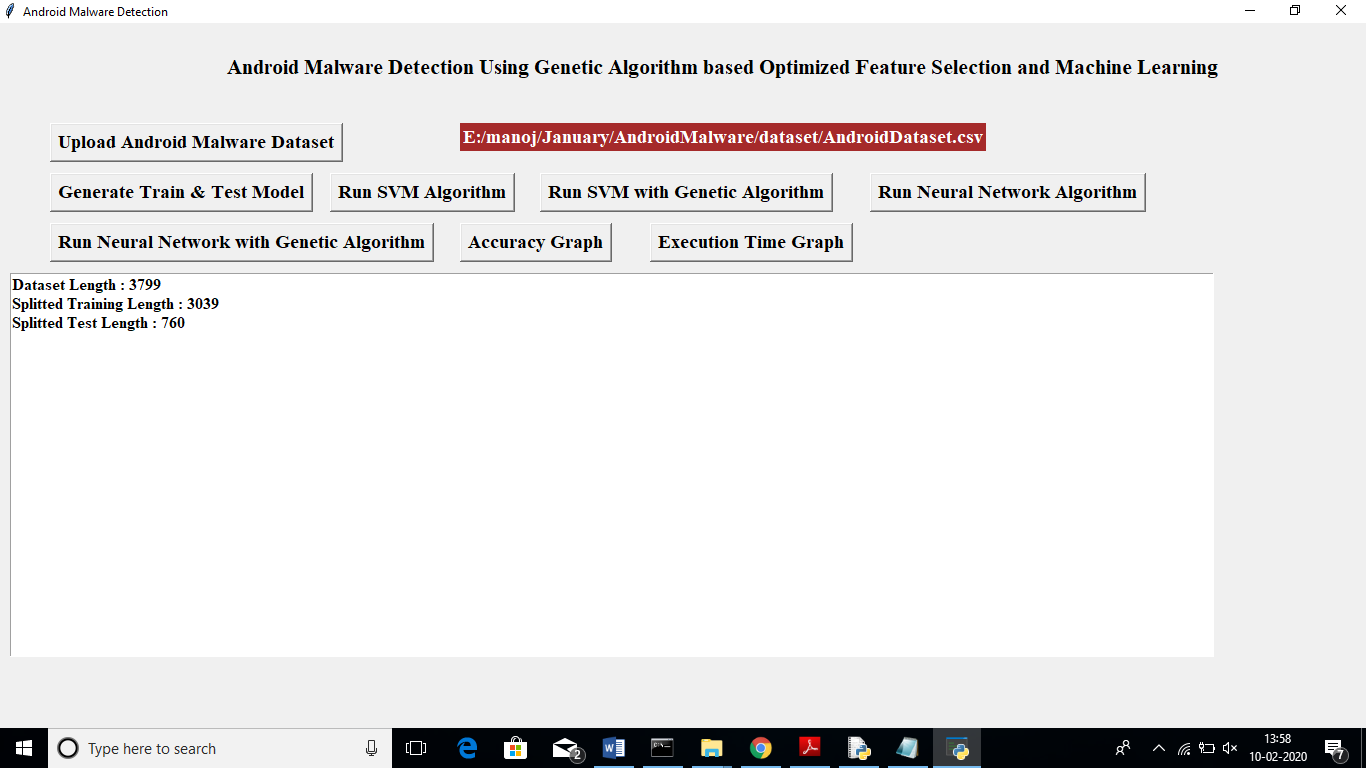
In above screen click on ‘Upload Android Malware Dataset’ button and upload dataset.



Screenshot 5.2 : uploading Android Data set

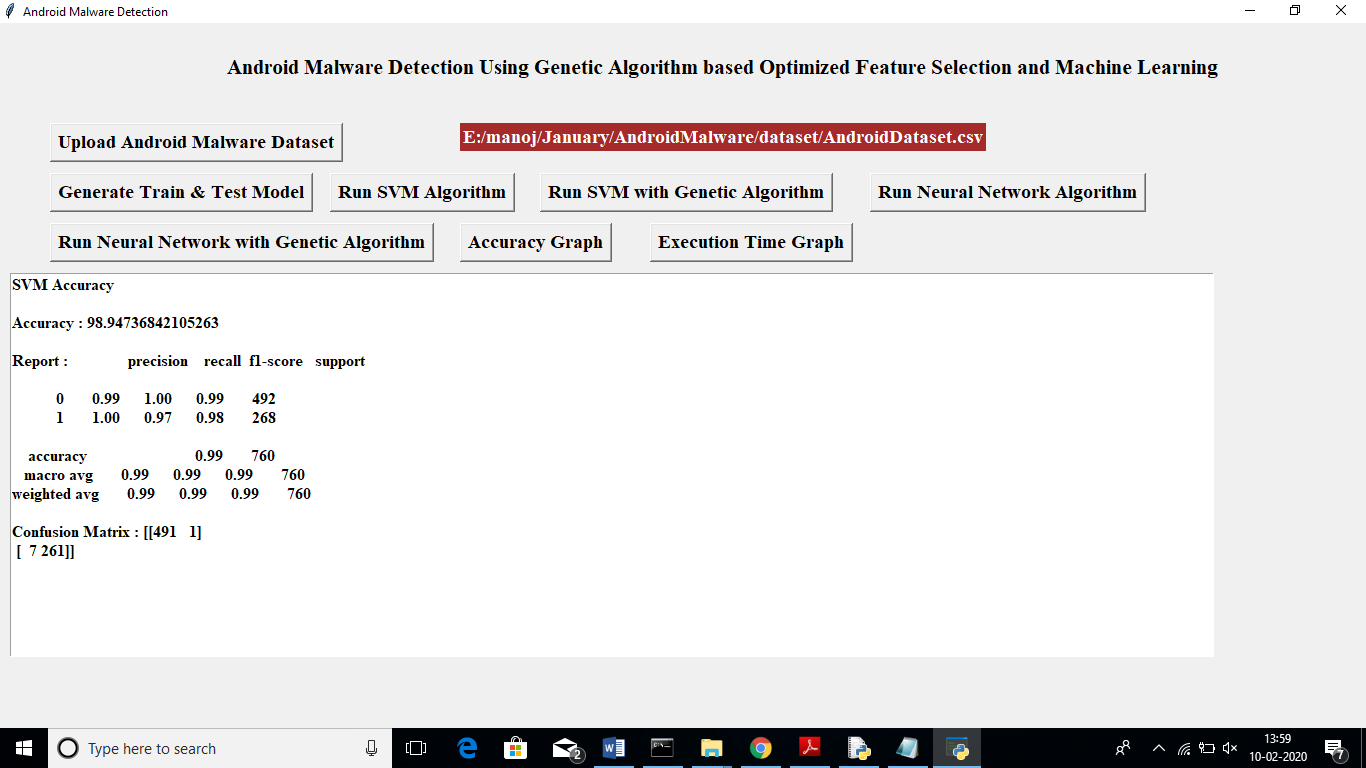
In above screen I am uploading ‘AndroidDataset.csv’ file and after upload will get below screen.

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.



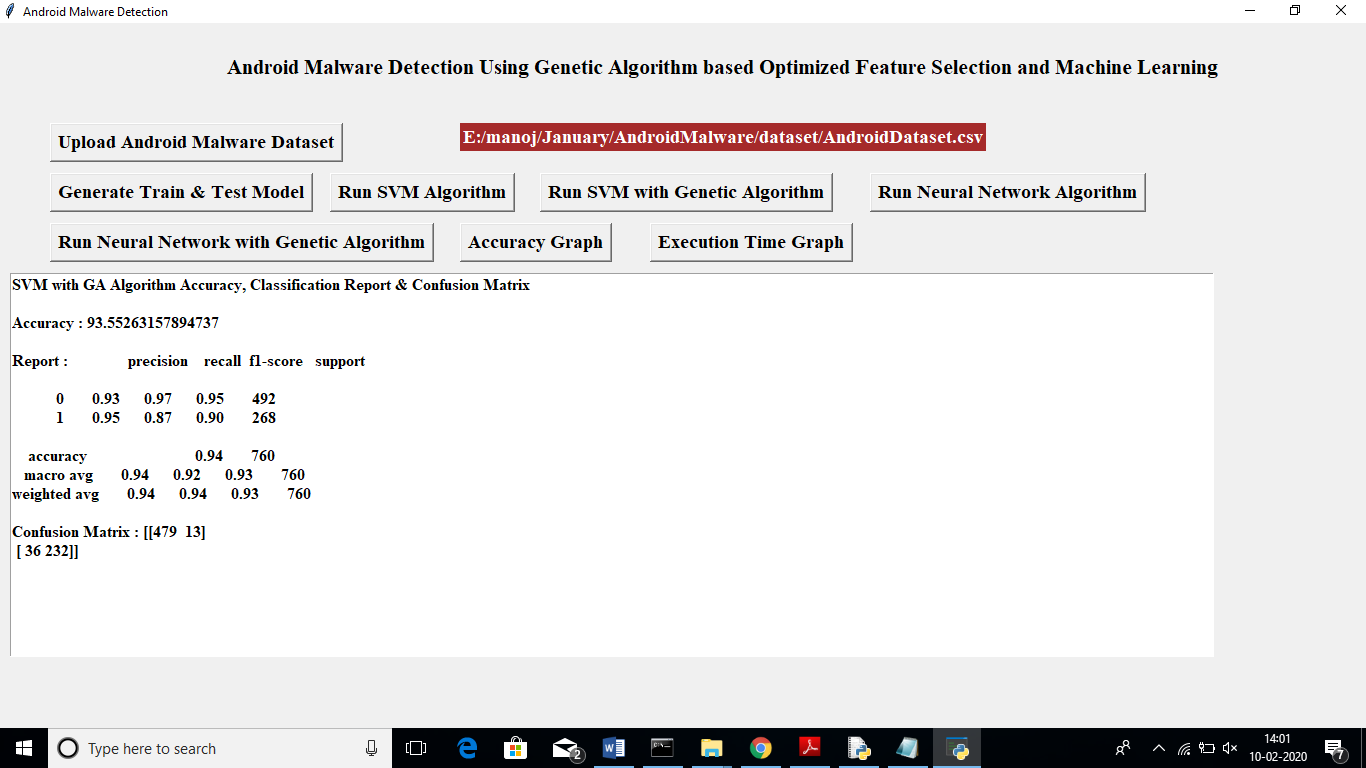
Screenshot 5.3 : After Generate Train And Testing data

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.



Screenshot 5.4 : After Running SVM Algorithm

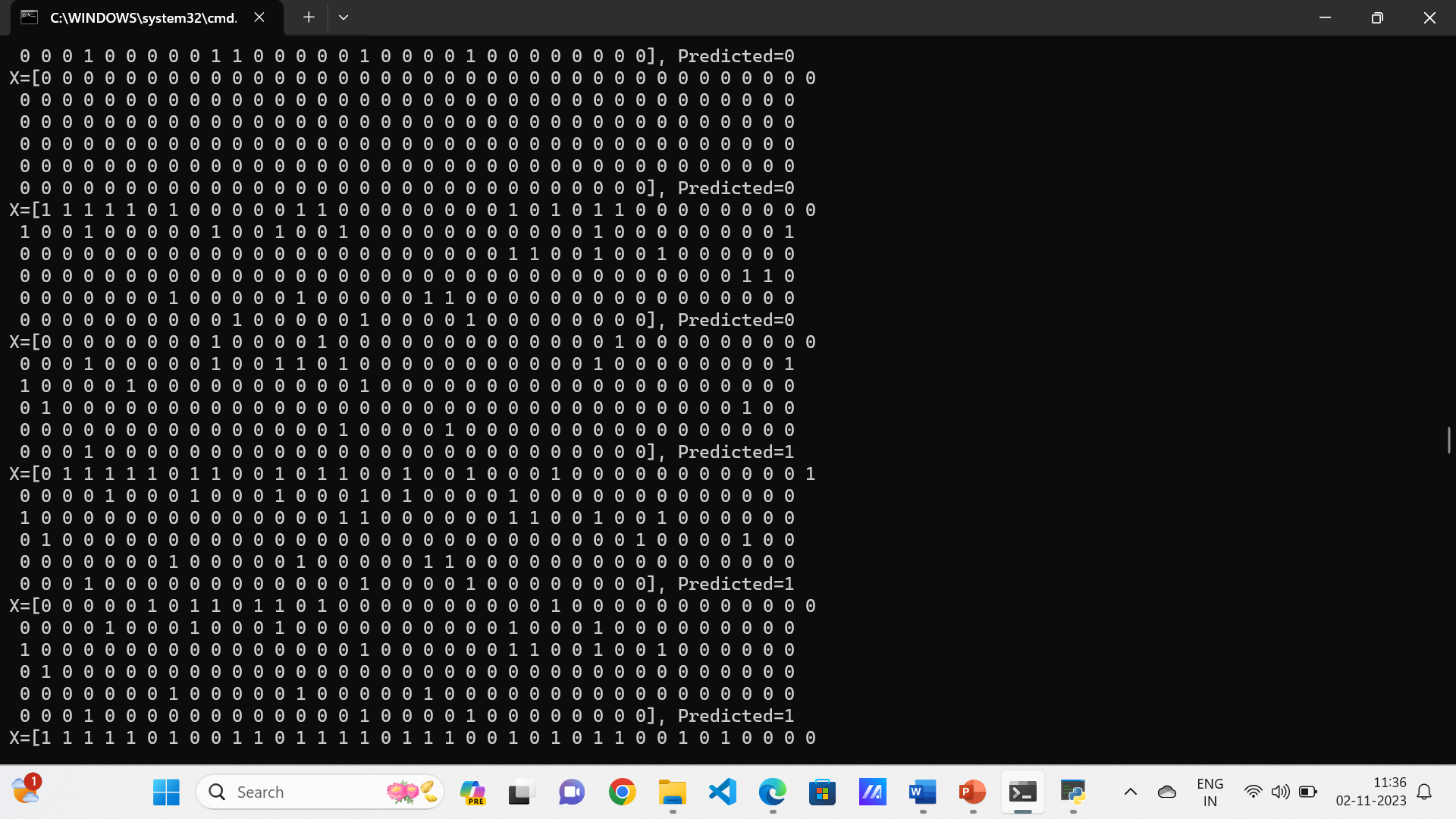
In above screen we got 98% accuracy for SVM and now click on ‘Run SVM with Genetic Algorithm’ button to choose optimize features and then run SVM on optimize features to get accuracy.



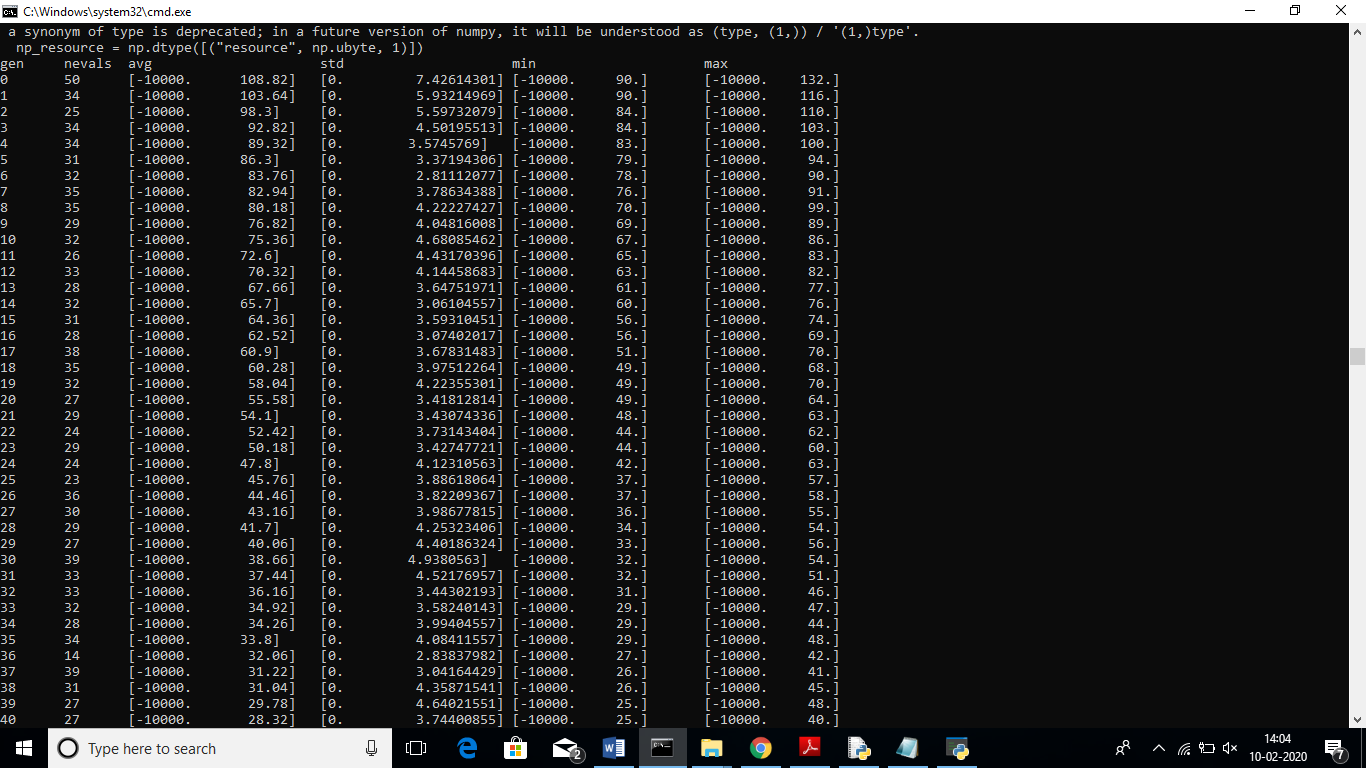
Screenshot 5.5 : After Running SVM with Genetic Algorithm

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).



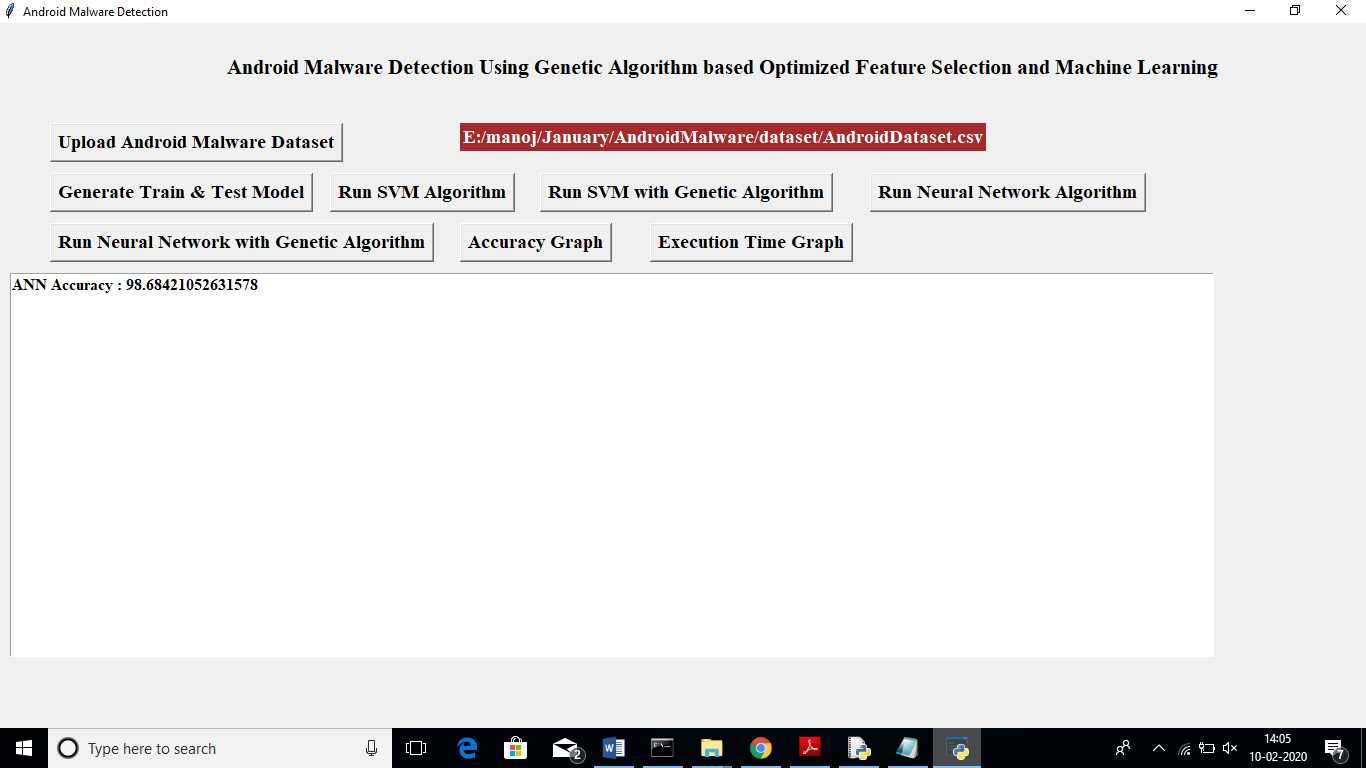
Screenshot 5.6 : Genetic algorithm prediction of dataset



Screenshot 5.7 : Genetic algorithm chooses 40 features from all dataset features

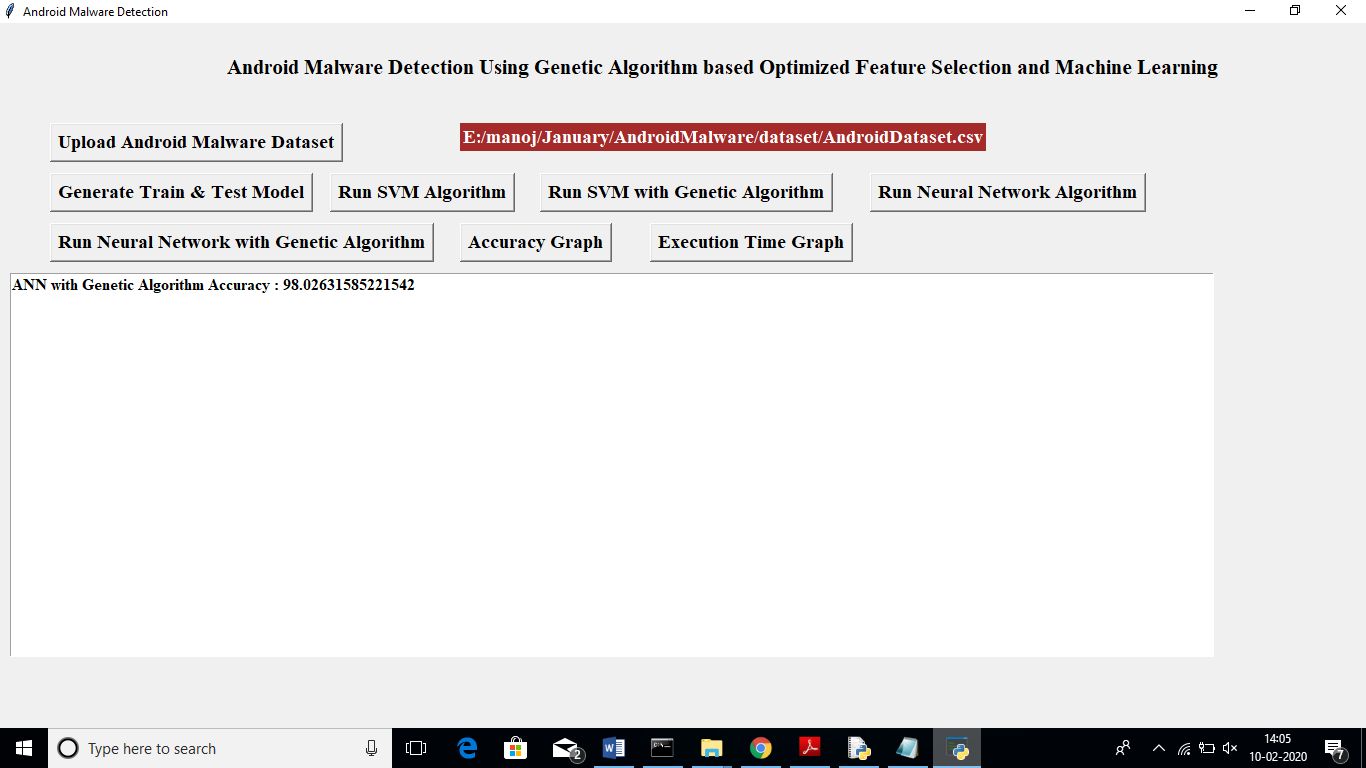
In above console we can see genetic algorithm chooses 40 features from all dataset features.

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



Screenshot 5.8 : After Running Neural Network Algorithm

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.



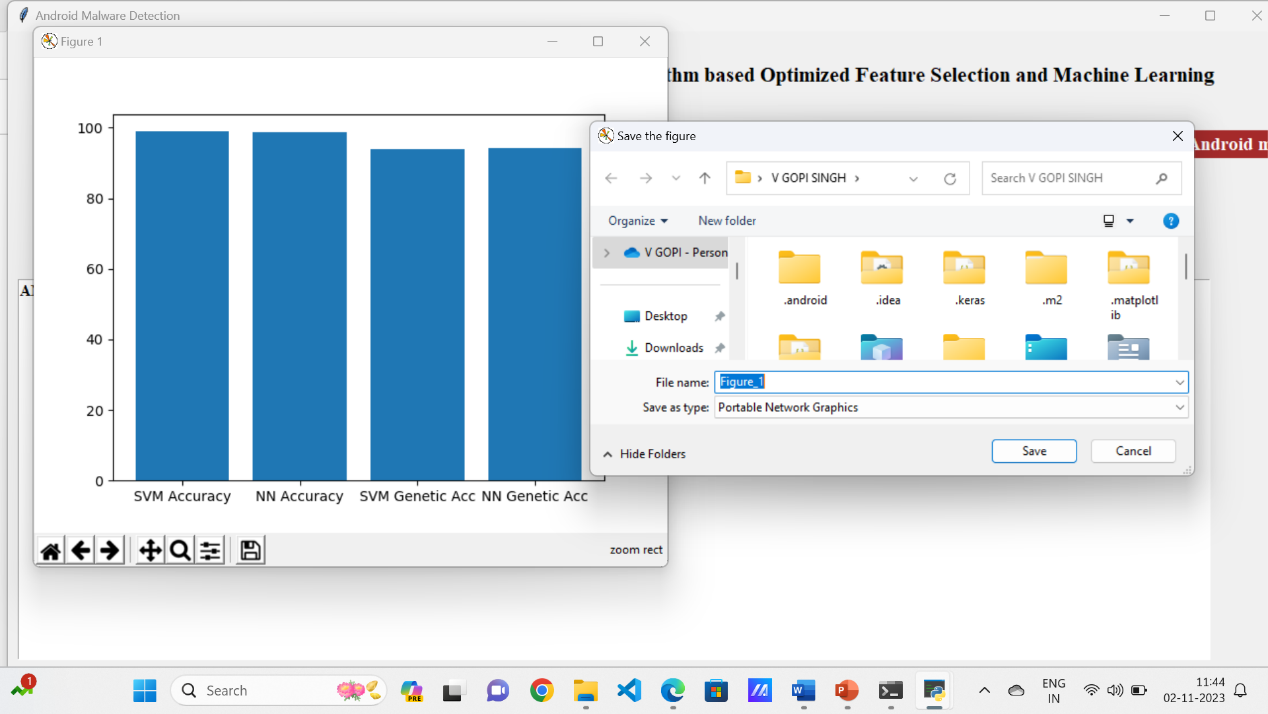
Screenshot 5.9 : After Running Neural Network with Genetic Algorithm

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

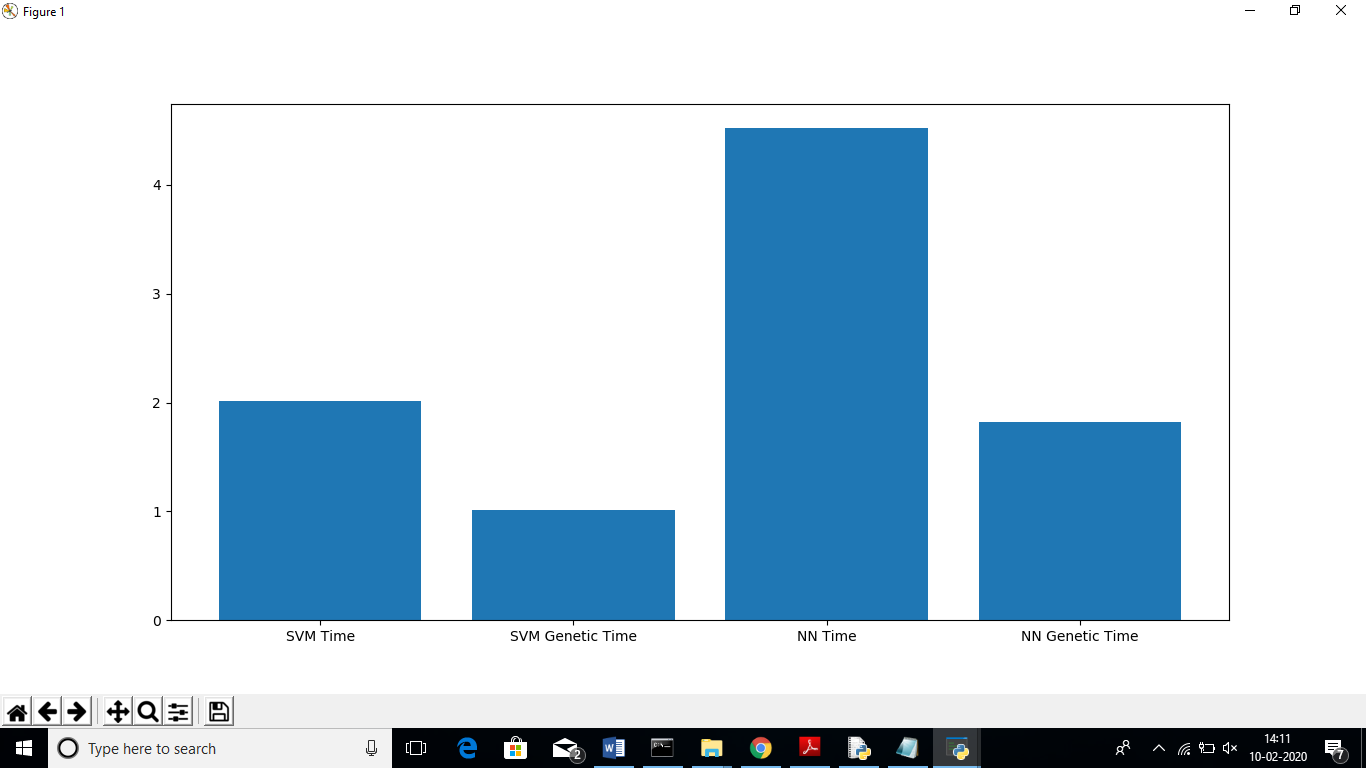


Screenshot 5.10 : Accuracy Graph

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.



Screenshot 5.11 : We can save Graph for future.



Screenshot 5.12 : Execution Time Graph

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.

**6. TESTING**

**6. TESTING**

**6.1 INTRODUCTION TO TESTING**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

**6.2 TYPES OF TESTING**

**6.2.1 UNIT TESTING**

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

**6.2.2 INTEGRATION TESTING**

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

**6.2.3 FUNCTIONAL TESTING**

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

Functional testing is centred on the following items:

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

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

Functions **:** identified functions must be exercised.

Output **:** identified classes of application outputs must be exercised.

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

**6.2.3.1 SYSTEM TEST**

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

**6.2.3.2 WHITE BOX TESTING**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**6.2.3.3 BLACKBOX TESTING**

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

**UNIT TESTING**

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

**TEST STRATEGY AND APPROACH**

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

**TEST OBJECTIVES**

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

**FEATURES TO BE TESTED**

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

**Test Results:**

All the test cases mentioned above passed successfully. No defects encountered.

**ACCEPTANCE TESTING**

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

**7. CONCLUSION**

**7. CONCLUSION & FUTURE SCOPE**

**7.1 PROJECT 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.

**7.2 FUTURE SCOPE**

From this project, 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.

1. **BIBLIOGRAPHY**

1. **BIBLIOGRAPHY**

**8.1 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.

**GITHUB LINK :-**

<https://github.com/Vgs26/ANDROID-MALWARE-DETECTION-USING-MACHINE-LEARNING-TECHNIQUES->