A

Major Project

On

ANDROID MALWARE DETECTION USING GENETIC ALGORITHM

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

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

P. LIKHITHA KRISHNAJA (187R1A05M4)
A. PRANAY (197R5A0515)
N. SUKRUTHA (197R5A0514)

Under the Guidance of

NAJEEMA AFRIN

(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.

2018-22

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled "**ANDROID MALWARE DETECTION USING GENETIC ALGORITHM"** being submitted by **P. LIKHITHA KRISHNAJA (187R1A05M4), A. PRANAY (197R5A0515), N. SUKRUTHA (197R5A0514)** in partial fulfillment 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 him/her under our guidance and supervision during the year 2021-22.

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

Najeema Afrin
Assistant Professor
INTERNAL GUIDE

Dr. A. Raji Reddy
DIRECTOR

Dr. K. Srujan Raju
HOD

EXTERNAL EXAMINER

Submitted for viva voice Examination held on _____

ACKNOWLEGDEMENT

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 **Mrs. Najeema Afrin,** 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 Project Review Committee (PRC) **Mr. A. Uday Kiran, Mr. J. Narasimha Rao, Dr. T. S. Mastan Rao, Mrs. G. Latha, Mr. A. Kiran Kumar** 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.

P. LIKHITHA KRISHNAJA (187R1A05M4)
A. PRANAY (197R5A0515)
N. SUKRUTHA (197R5A0514)

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.

LIST OF FIGURES/TABLES

List of Screenshots

TABLE OF CONTENTS

1. INTRODUCTION

1. INTRODUCTION

1.1 PROJECT SCOPE

As the use of smartphones increases, Android, as a Linux-based open-source mobile operating system (OS), has become the most popular mobile OS in time. Due to the widespread use of Android, malware developers mostly target Android devices and users. Malware detection systems to be developed for Android devices are important for this reason. Machine learning methods are being increasingly used for detection and analysis of Android malware. This study presents a method for detecting Android malware using feature selection with genetic algorithm.

1.2 PROJECT PURPOSE

The purpose of this study is 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.

1.3 PROJECT FEATURES

The main feature of this system is to propose a general and effective approach to detect the malicious applications for android operating system. 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.

2. SYSTEM ANALYSIS

2. SYSTEM ANALYSIS

SYSTEM ANALYSIS

System Analysis is the important phase in the system development process. The System is studied to the minute details and analyzed. Analysis is the process of finding the best solution to the problem. System analysis is the process by which we learn about the existing problems, define objects and requirements, and evaluate the solutions. It is the way of thinking about the organization and the problem it involves, a set of technologies that helps in solving these problems. Feasibility study plays an important role in system analysis which gives the target for design and development.

2.1 PROBLEM DEFINITION

This project is primarily concerned with improving the accuracy of malware detection and the time required to develop the model. The suggested approach attempts to use a Genetic Algorithm to obtain the most optimal function subset that can be used to inform machine studying algorithms in the most effective manner.

- The major objective is to detect android malwares in each dataset.
- To create a software which uses the given dataset to train & test the available algorithms and detect the Malware applications if any in the given new data.
- Comparison between the three algorithms i.e. SVM, Neural Network and Genetic Algorithm for more precise evaluation.

2.2 EXISTING SYSTEM

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. Due to its open-source nature and popularity, malware writers are increasingly focusing on developing malicious applications for Android operating system. Despite 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.

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.

The 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

2.2.1 LIMITATIONS OF EXISTING SYSTEM

- Storing of large amounts of data that contains a lot of information about malicious application is posing a challenge for the app users.
- Sometimes the data is entered manually, and humans can make mistakes, so there are chances of incorrect data being entered in the dataset which can lead to inaccurate results while analyzing the data.
- In such a large dataset, there is always a chance of some fields containing missing values, these missing values can make the data noisy and thus we must take appropriate measures to remove inconsistency from the datasets.
- Lack of sufficient analytical support to back up their data.
- Due to its opensource nature and popularity, malware writers are increasingly focusing on developing malicious applications for Android operating system.

2.3 PROPOSED SYSTEM

- The proposed system implements machine algorithm to detect malicious app data in android operating system using genetic algorithm's optimized feature selection.
- 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 feature vector 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.

2.3.1 ADVANTAGES OF THE PROPOSED SYSTEM

The system is very simple in design and to implement. The system requires very low system resources, and the system will work in almost all configurations. It has got following features

- Better services
- Ensure data accuracies.
- Greater efficiency.
- Security
- Minimum time needed for the various processing.
- 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.

2.4 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. Three key considerations involved in the feasibility analysis are

- Economic Feasibility
- Technical Feasibility
- Social Feasibility

2.4.1 ECONOMIC FEASIBILITY

The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on project, which will give best, return at the earliest. One of the factors, which affect the development of a new system, is the cost it would require.

The following are some of the important financial questions asked during preliminary investigation:

- The costs conduct a full system investigation.
- The cost of the hardware and software.
- The benefits in the form of reduced costs or fewer costly errors.

Since the system is developed as part of project work, there is no manual cost to spend for the proposed system. Also, all the resources are already available, it gives an indication of the system is economically possible for development.

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. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

2.4.3 BEHAVIORAL FEASIBILITY

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

2.5 HARDWARE & SOFTWARE REQUIREMENTS

2.5.1 HARDWARE REQUIREMENTS:

Hardware interfaces specifies the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements.

- System : Intel i5
- Hard Disk : 30GB and above
- Ram : 4GB and above

2.5.2 SOFTWARE REQUIREMENTS:

Software Requirements specifies the logical characteristics of each interface and software components of the system. The following are some software requirements.

- Operating system : Windows 10
- Programming language : Python 3.8
- Environment : Anaconda
- Tool : Jupyter Notebook

3. ARCHITECTURE

3. ARCHITECTURE

3.1 PROJECT ARCITECTURE

This project architecture shows the procedure followed for breed detection using machine learning, starting from input to final prediction.

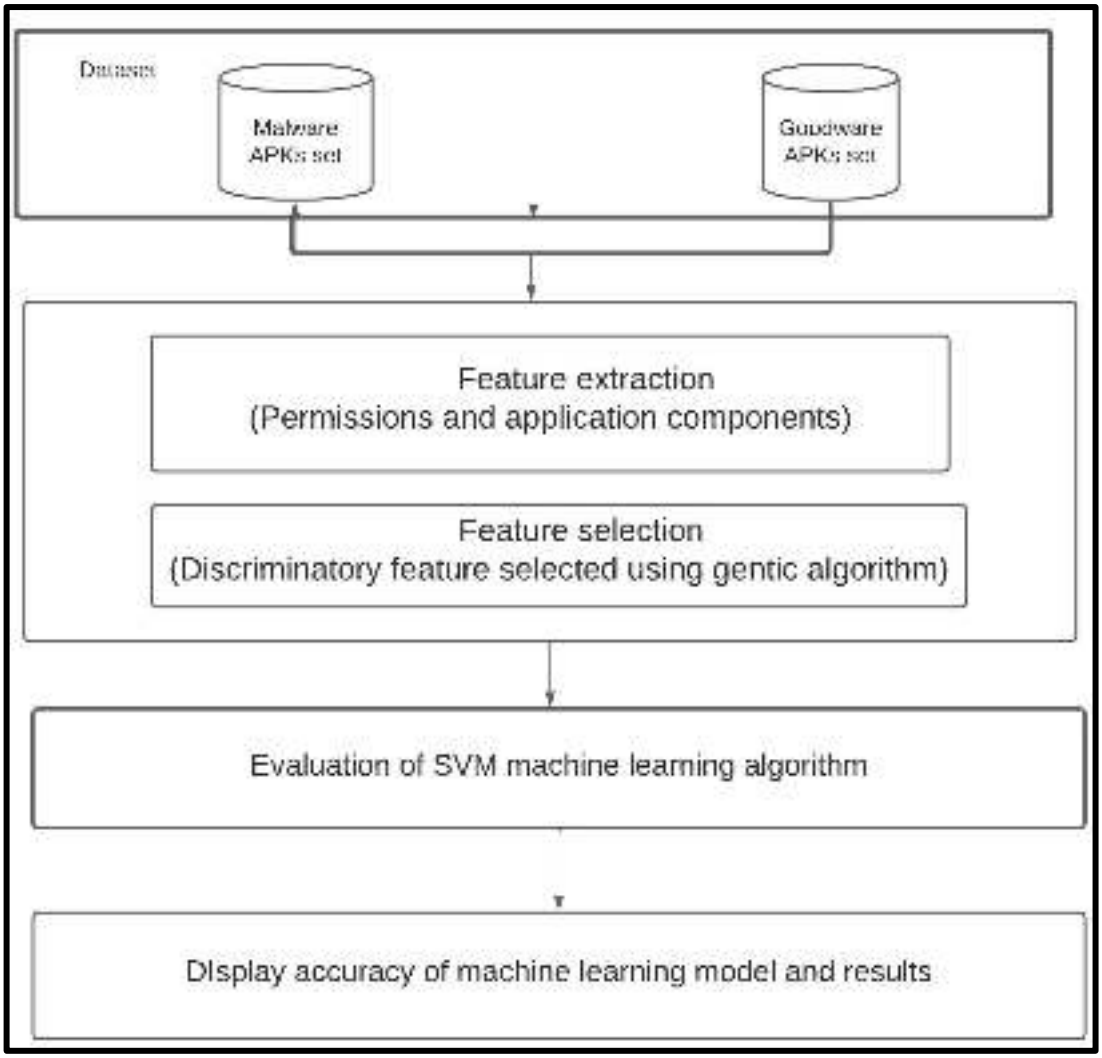


Figure 3.1: Project Architecture of Android Malware Detection using Genetic Algorithm

3.2 MODULES DESCRIPTION

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.

3.2.1 DATA PREPROCESSING

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 feature vector with class labels as Malware and Goodware represented by 0 and 1 respectively in CSV format.

3.2.2 FEATURES EXTRACTION AND SELECTION

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 proposed methodology, basically involving two units: feature extraction using Androguard tool and feature selection using Genetic Algorithm. Finally, the selected features are fed as input to machine learning algorithms for evaluation purpose.

A. Reverse-Engineering of Android APKs

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.

B. Feature Vector

Features are extracted and mapped to a feature vector as follows:

App Components: The counts of App components such as Activity, Services, Content Providers and Broadcast Receivers are used as a feature vector.

Permissions: The permissions feature-set are mapped to a |S| dimensional vector space such that a dimension is set to 1 if the app x contains the feature and 0 otherwise. In this way, a vector $\psi(x)$ is constructed for each feature extracted from app x with the respective dimension is set to 1 and all other dimensions to 0. It can be summarized in equation (1): ψ: X {0;1} |S|

3.2.3 Discriminatory Feature Selection

In malware detection, selecting most significant features is an important step as it has a significant impact on quality of experimental results. Also, working on low-dimensional feature vector consisting of only discriminatory features will help in reducing computational complexity of learning classifier. The CSV consisting of all features is fed into Genetic algorithm which gives best subset of features for the machine learning based classifier.

Features selected are represented by binary form called chromosomes such that if the feature is included it is represented by 1 and if it is excluded it is represented by 0 in the chromosome. The genetic algorithm maintains a subset of features or chromosome called population along with their fitness scores such that chromosome with better fitness scores are given more chance to reproduce.

The fitness function of genetic algorithm is defined such that the chromosome that gives high accuracy on the machine learning based classifier is assigned a larger value in comparison to features that give lower accuracy for it. The chromosomes with best fitness score are selected as parent to produce next generation of offspring using the process of crossover and mutation.

The steps involved in feature selection using Genetic Algorithm can be summarized as below:
Step 1: Initialize the algorithm using feature subsets which are binary encoded such that if the feature is included it is represented by 1 and if it is excluded it is represented by 0 in the chromosome.
Step 2: Start the algorithm defining an initial set of population generated randomly.
Step 3: Assign a fitness score calculated by the defined fitness function for genetic algorithm.
Step 4: Selection of Parents: Chromosomes with good fitness scores are given preference over others to produce next generation of off-springs.
Step 5: Perform crossover and mutation operations on the selected parents with the given probability of crossover and mutation for generation of off-springs. Repeat the Steps 3 to 5 iteratively till the convergence is met and fittest chromosome from population, that is, the optimal feature subset is resulted.
Repeat the Steps 3 to 5 iteratively till the convergence is met and fittest chromosome from population, that is, the optimal feature subset is resulted.

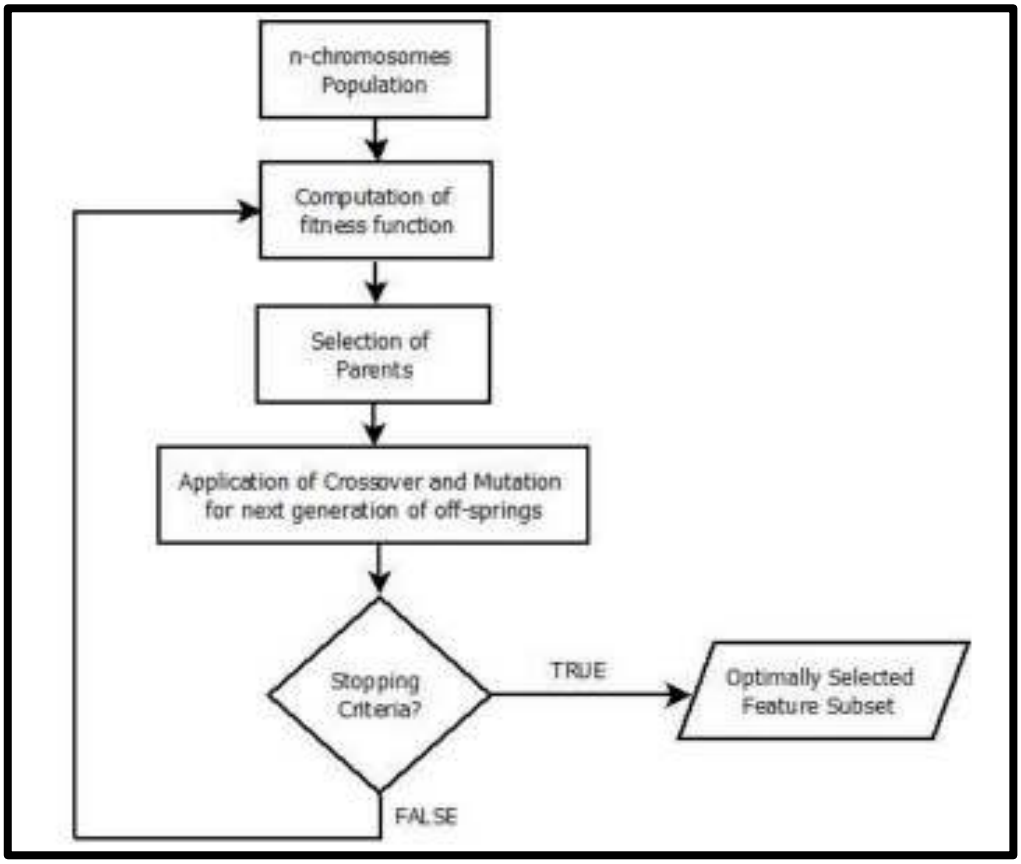


Figure 3.2.3 Feature Selection using Genetic Algorithm

3.2.4 MACHINE LEARNING BASED CLASSIFICATION

Given in to the ever-increasing variants of Android Malware posing zero-day threat, machine learning based techniques are being preferred over traditional signature-based approach which required regular update of signature database. The selected features using Genetic Algorithm are used to train and test the classifiers with following algorithms:

Support Vector Machine Algorithm (SVM)

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future.

This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:

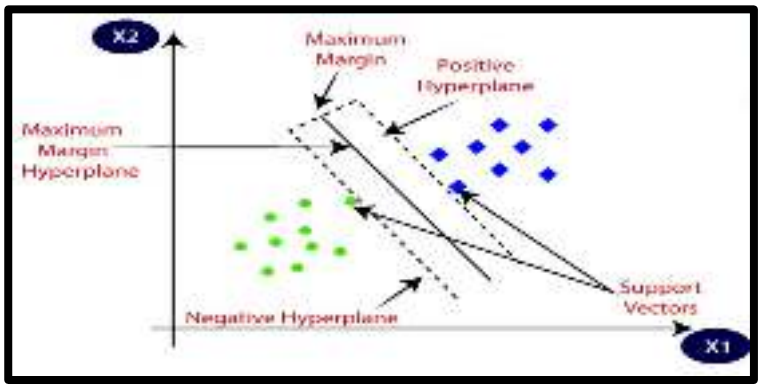


Figure 3.2.4.1 Support Vector Machine Algorithm

SVM algorithm can be used for Face detection, image classification, text categorization, etc.

Types of SVM -SVM can be of two types: 5

Linear SVM: Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

Non-linear SVM: Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Hyperplane and Support Vectors in the SVM algorithm:

Hyperplane: There can be multiple lines/decision boundaries to segregate the classes in ndimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM. The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane. We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

Support Vectors: The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

Neural Networks Algorithm (NN)

It is a procedure learning system that uses a network of functions to grasp and translate an information input of 1 kind into the specified output, sometimes in another kind. The thought if the unreal neural network was impressed by human biology and therefore the method neurons of the human brain along to grasp inputs from human senses.

Neural networks are only one of the numerous tools and approaches employed in machine learning algorithms. The neural network itself is also used as a bit in many various machine learning algorithms to method advanced inputs into areas that computers will perceive.

Neural networks are inspired by the biological neural networks in the brain, or we can say the nervous system. It has generated a lot of excitement, and research is still going on this subset of Machine Learning in the industry. The basic computational unit of a neural network is a neuron or node. It receives values from other neurons and computes the output. Each node/neuron is associated with weight(w). This weight is given as per the relative importance of that neuron or node. So, if we take f as the node function, then the node function f will provide output as shown below:

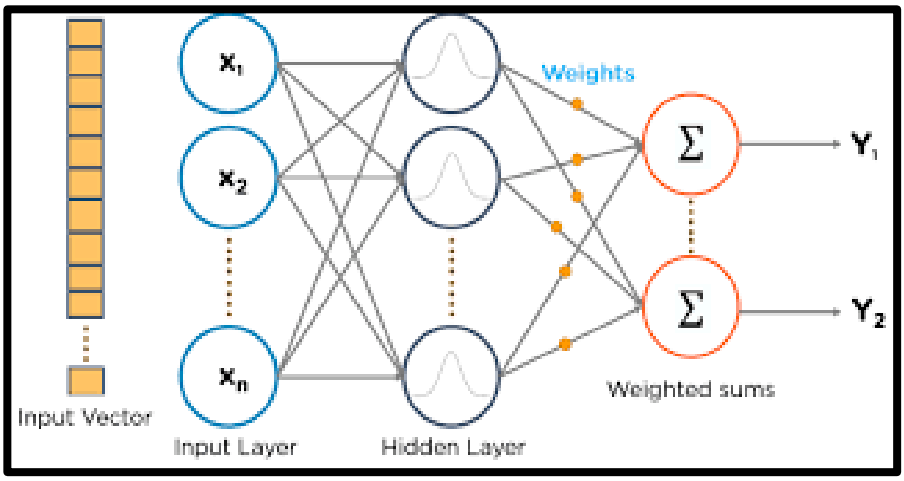


Figure 3.2.4.2 Neural Networks machine learning model

Output of neuron(Y) = f (w1.X1 +w2.X2 +b)

Where w1 and w2 are weight, X1 and X2 are numerical inputs, whereas b is the bias.

The above function f is a non-linear function also called the activation function. Its basic purpose is to introduce non-linearity as almost all real-world data is non-linear, and we want neurons to learn these representations.

3.2.5 DISPLAY ACCURACY RESULTS

After selection of machine learning model and uploading application APKs on clicking predict button we obtain accuracy results of predicted android malware applications.

3.3 USE CASE DIAGRAM

In the use case diagram, we have basically two actors who are the user and the administrator. The user has the rights to login, access to resources and to view the crime details. Whereas the administrator has the login, access to resources of the users and the right to update and remove the crime details, and he can also view the user files.

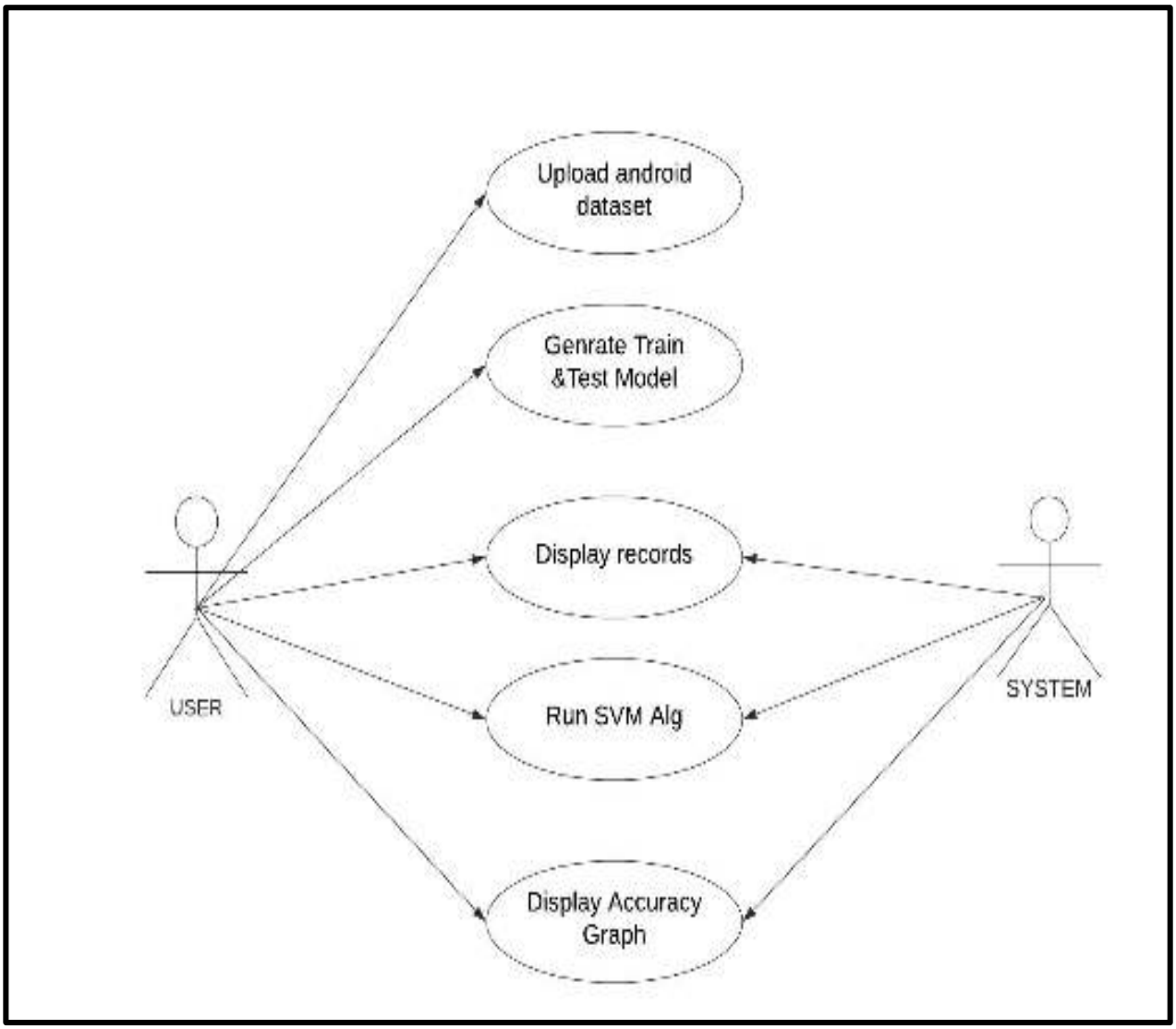


Figure 3.3: Use Case Diagram for Android Malware Detection Using Genetic Algorithm

3.4 CLASS DIAGRAM

Class Diagram is a collection of classes and objects.

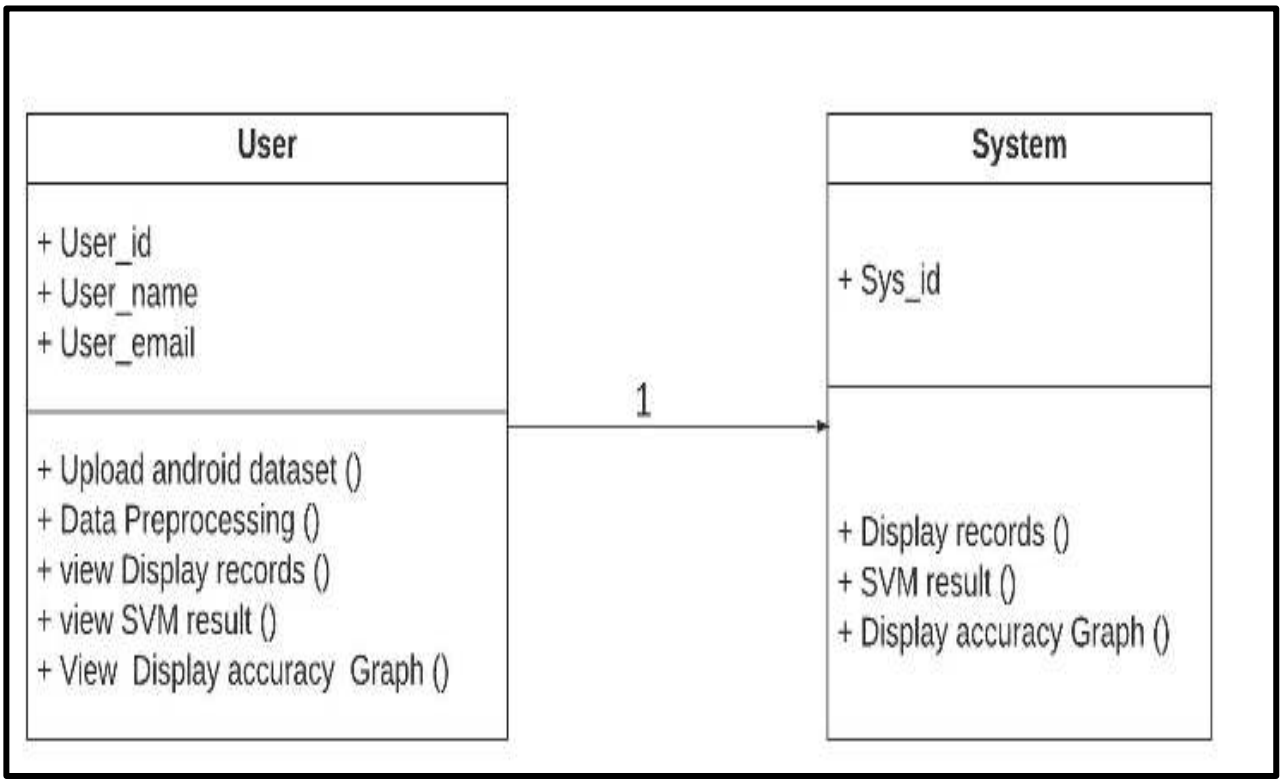


Figure 3.4: Class Diagram for Android Malware Detection using Genetic Algorithm

3.5 SEQUENCE DIAGRAM

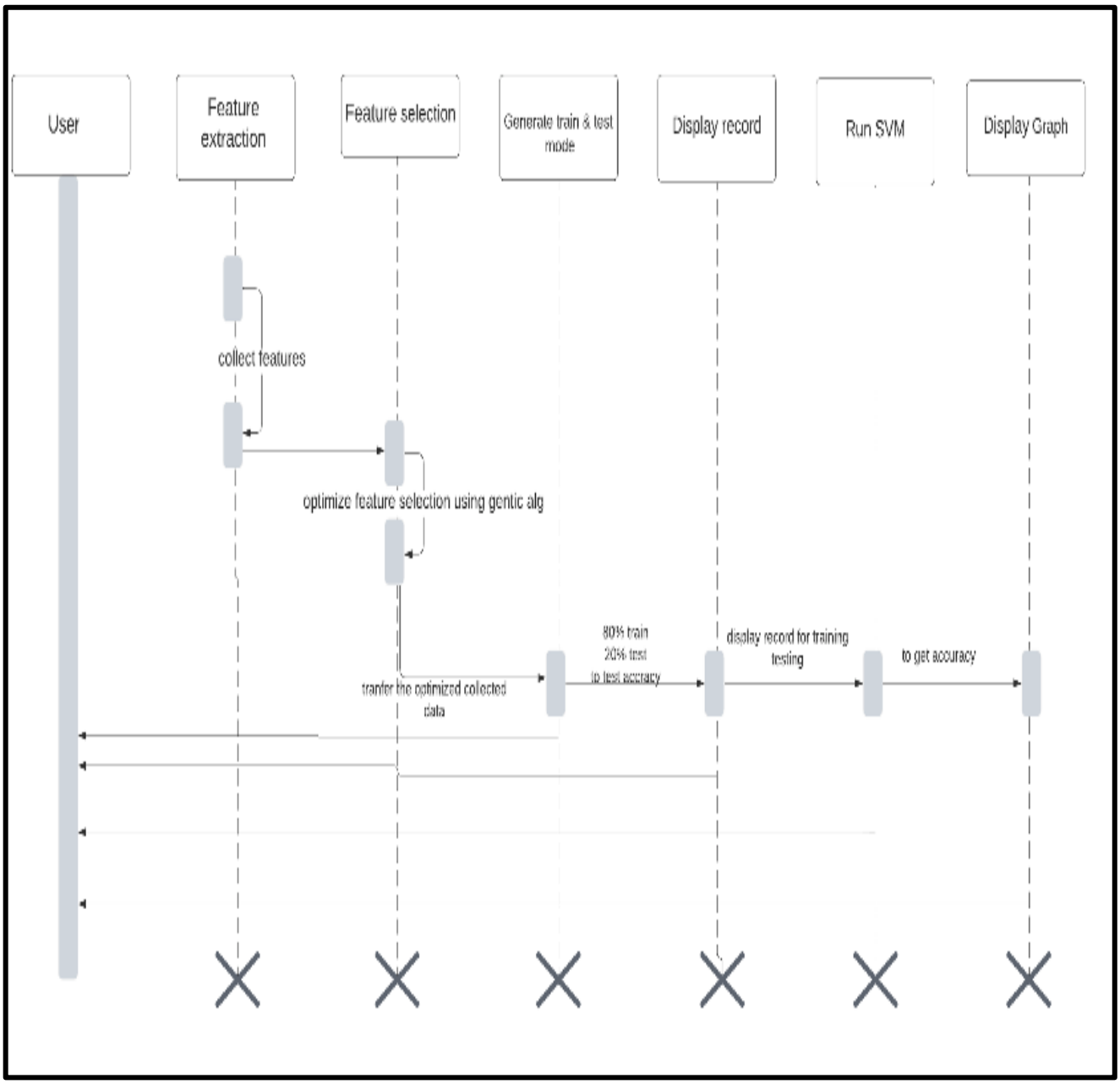


Figure 3.5: Sequence Diagram of Android Malware Detection using Genetic Algorithm

3.6 ACTIVITY DIAGRAM

It describes about flow of activity states.

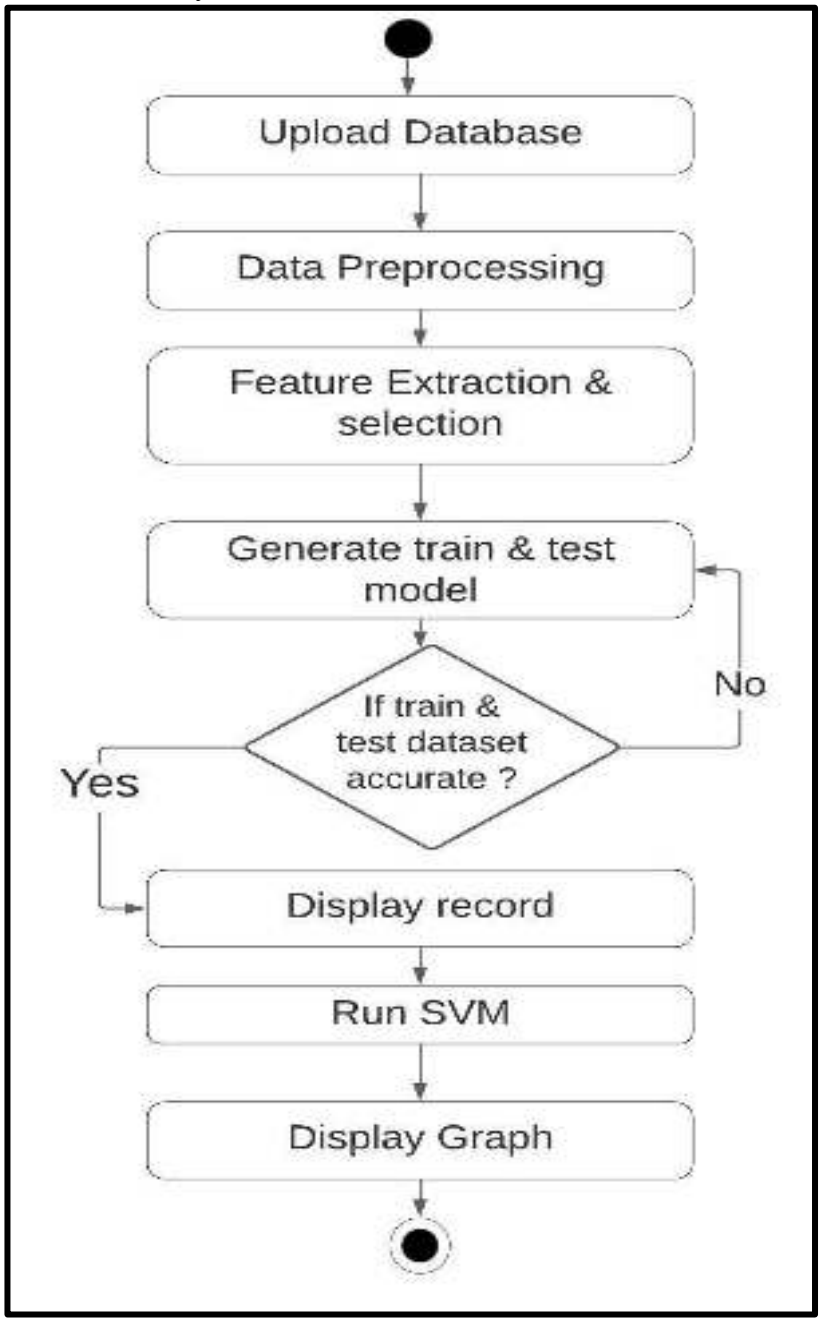


Figure 3.6: Activity Diagram for Android Malware Detection using Genetic Algorithm

3.7 DATAFLOW DIAGRAM

It describes about flow of activity data.

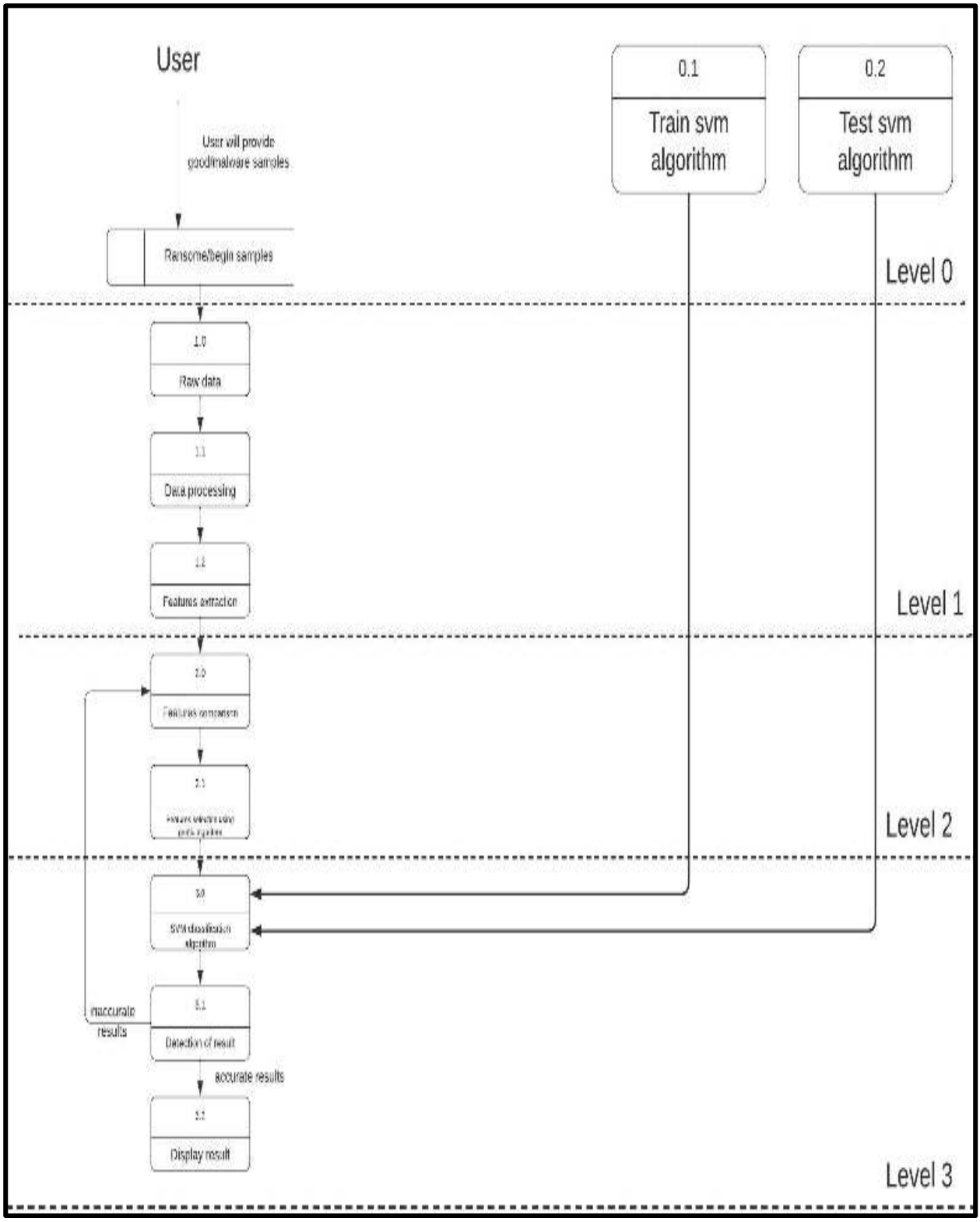


Figure 3.7: Dataflow Diagram for Android Malware Detection using Genetic Algorithm

4. IMPLEMENTATION

4. IMPLEMENTATION

4.1 SAMPLE CODE

A) APP.PY

```
from flask import Flask, render_template, request, redirect, url_for, flash
from werkzeug.utils import secure_filename
import os
import classifier
app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = './static/upload/'
app.config['SECRET_KEY'] = 'd3Y5d5nJkU6CdwY'
if os.path.exists(app.config['UPLOAD_FOLDER']):
    print("directory exists")
else:
    os.makedirs(app.config['UPLOAD_FOLDER'])
    print("directory created")
@app.route("/", methods=["GET", "POST"])
def home():
    algorithms = {'Neural Network': '92.26 %', 'Support Vector Classifier': '89 %'}
    result, accuracy, name, sdk, size = '', '', '', '', ''
    if request.method == "POST":
        if 'file' not in request.files:
            flash('No file part')
            return redirect(request.url)
        file = request.files['file']
        # if user does not select file, browser also
        # submit an empty part without filename
        if file.filename == '':
            flash('No selected file')
            return redirect(request.url)
        if file and file.filename.endswith('.apk'):
            filename = secure_filename(file.filename)
            print(filename)
            file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
            if request.form['algorithm'] == 'Neural Network':
                accuracy = algorithms['Neural Network']
                result, name, sdk, size = classifier.classify(os.path.join(app.config['UPLOAD_FOLDER'],
filename), 0)
            elif request.form['algorithm'] == 'Support Vector Classifier':
                accuracy = algorithms['Support Vector Classifier']
                result, name, sdk, size = classifier.classify(os.path.join(app.config['UPLOAD_FOLDER'],
filename), 1)
    return render_template("index.html", result=result, algorithms=algorithms.keys(),
accuracy=accuracy, name=name,
                           sdk=sdk, size=size)
if __name__ == "__main__":  # on running python app.py
    app.run(debug=False)  # run the flask app
```

B) CLASSIFIER.PY

```
import os
import pickle
import numpy as np
from keras.models import load_model
from androguard.core.bytecodes.apk import APK
from genetic_algorithm import GeneticSelector
class CustomUnpickler(pickle.Unpickler):
    """ https://stackoverflow.com/questions/27732354/unable-to-load-files-using-pickle-and-multiple-modules"""

    def find_class(self, module, name):
        try:
            return super().find_class(__name__, name)
        except AttributeError:
            return super().find_class(module, name)
sel = CustomUnpickler(open('./static/models/ga.pkl', 'rb')).load()
permissions = []
with open('./static/permissions.txt', 'r') as f:
    content = f.readlines()
    for line in content:
        cur_perm = line[:-1]
        permissions.append(cur_perm)
def classify(file, ch):
    vector = {}
    result = 0
    name, sdk, size = 'unknown', 'unknown', 'unknown'
    app = APK(file)
    perm = app.get_permissions()
    name, sdk, size = meta_fetch(file)
    for p in permissions:
        if p in perm:
            vector[p] = 1
        else:
            vector[p] = 0
    data = [v for v in vector.values()]
    data = np.array(data)
    if ch == 0:
        ANN = load_model('static/models/ANN.h5')
        #print(data)
        result = ANN.predict([data[sel.support_].tolist()])
        print(result)
        if result < 0.02:
            # return 'Benign(safe)'
            result = 'Benign(safe)'
        else:
            # return 'Malware'
            result = 'Malware'
```

```
if ch == 1:
        SVC = pickle.load(open('static/models/svc_ga.pkl', 'rb'))
        result = SVC.predict([data[sel.support_]])
        if result == 'benign':
            result = 'Benign(safe)'
        else:
            result = 'Malware'
  return result, name, sdk, size

def meta_fetch(apk):
    app = APK(apk)
    return app.get_app_name(), app.get_target_sdk_version(), str(round(os.stat(apk).st_size / (1024 *
1024), 2)) + ' MB'
```

C) GENETIC_ALOGIRTHM.PY

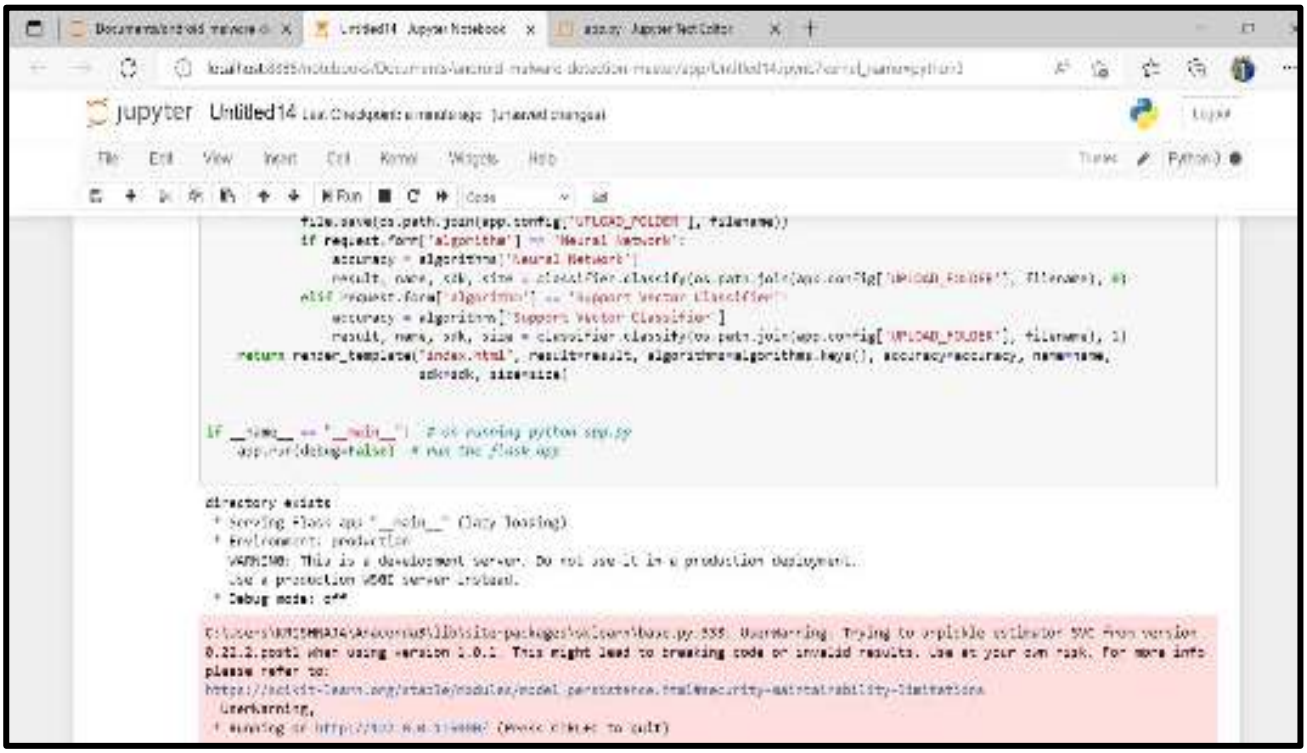
```
import random
import numpy as np
import matplotlib.pyplot as plt

from sklearn.model_selection import cross_val_score
class GeneticSelector:
    def __init__(self, estimator, n_gen, size, n_best, n_rand,
                 n_children, mutation_rate):
        # Estimator
        self.estimator = estimator
        # Number of generations
        self.n_gen = n_gen
        # Number of chromosomes in population
        self.size = size
        # Number of best chromosomes to select
        self.n_best = n_best
        # Number of random chromosomes to select
        self.n_rand = n_rand
        # Number of children created during crossover
        self.n_children = n_children
        # Probablity of chromosome mutation
        self.mutation_rate = mutation_rate
        if int((self.n_best + self.n_rand) / 2) * self.n_children != self.size:
            raise ValueError("The population size is not stable.")

    def initilize(self):
        population = []
        for i in range(self.size):
            chromosome = np.ones(self.n_features, dtype=np.bool)
            mask = np.random.rand(len(chromosome)) < 0.3
            chromosome[mask] = False
            population.append(chromosome)
        return population
```

```
def fitness(self, population):
        X, y = self.dataset
        scores = []
        for chromosome in population:
            score = -1.0 * np.mean(cross_val_score(self.estimator, X[:, chromosome], y,
                                        cv=5,
                                        scoring="neg_mean_squared_error"))
            scores.append(score)
        scores, population = np.array(scores), np.array(population)
        inds = np.argsort(scores)
        return list(scores[inds]), list(population[inds, :])

    def select(self, population_sorted):
        population_next = []
        for i in range(self.n_best):
            population_next.append(population_sorted[i])
        for i in range(self.n_rand):
            population_next.append(random.choice(population_sorted))
        random.shuffle(population_next)
        return population_next

    def crossover(self, population):
        population_next = []
        for i in range(int(len(population) / 2)):
            for j in range(self.n_children):
                chromosome1, chromosome2 = population[i], population[len(population) - 1 - i]
                child = chromosome1
                mask = np.random.rand(len(child)) > 0.5
                child[mask] = chromosome2[mask]
                population_next.append(child)
        return population_next
    def mutate(self, population):
        population_next = []
        for i in range(len(population)):
            chromosome = population[i]
            if random.random() < self.mutation_rate:
                mask = np.random.rand(len(chromosome)) < 0.05
                chromosome[mask] = False
            population_next.append(chromosome)
        return population_next

    def generate(self, population):
        # Selection, crossover and mutation
        scores_sorted, population_sorted = self.fitness(population)
        population = self.select(population_sorted)
        population = self.crossover(population)
        population = self.mutate(population)
        # History
        self.chromosomes_best.append(population_sorted[0])
        self.scores_best.append(scores_sorted[0])
        self.scores_avg.append(np.mean(scores_sorted)
        return population
```

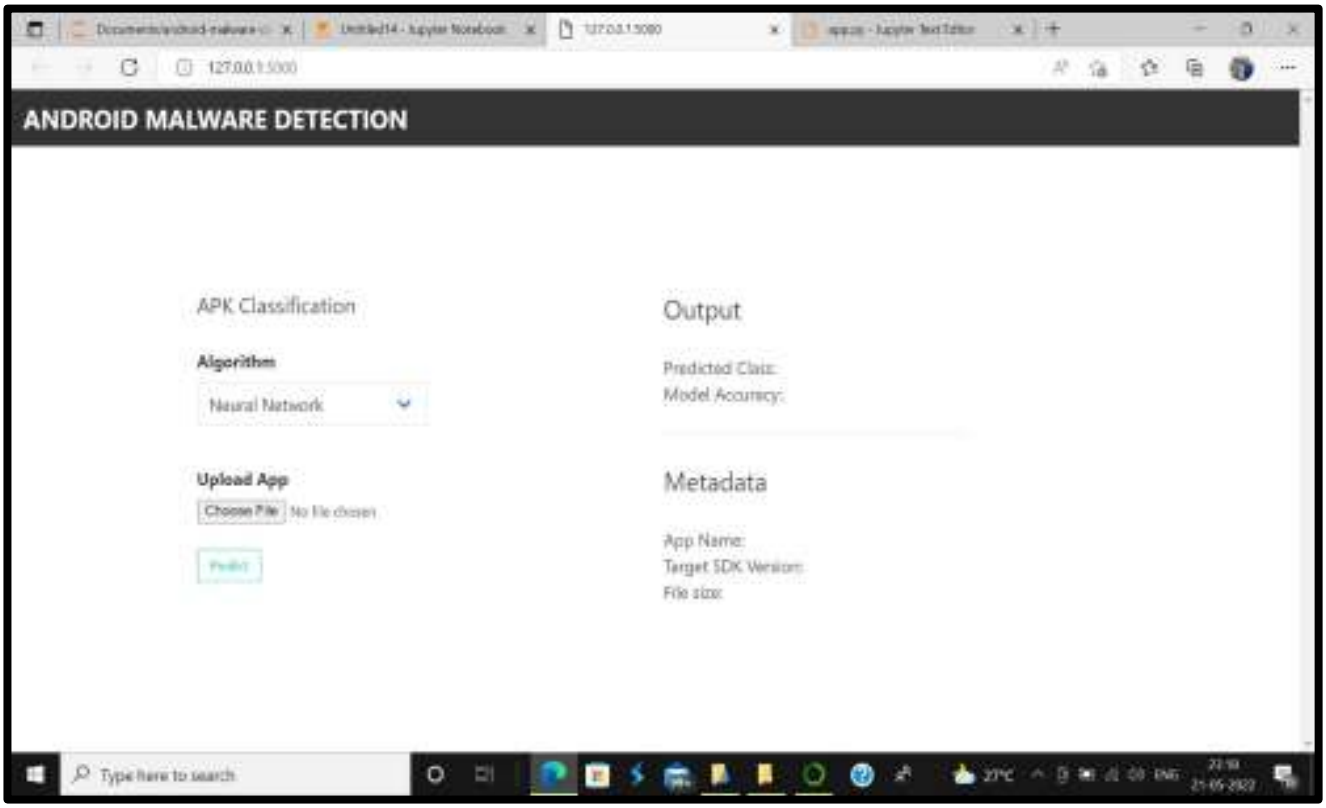
```
def fit(self, X, y):
        self.chromosomes_best = []
        self.scores_best, self.scores_avg = [], []

        self.dataset = X, y
        self.n_features = X.shape[1]
        g = 1
        population = self.initilize()
        for i in range(self.n_gen):
            population = self.generate(population)
            print('generation:', g)
            g += 1
        return self

    @property
    def support_(self):
        return self.chromosomes_best[-1]

    def plot_scores(self):
        plt.plot(self.scores_best, label='Best')
        plt.plot(self.scores_avg, label='Average')
        plt.legend()
        plt.ylabel('Scores')
        plt.xlabel('Generation')
        plt.show()
```

5. RESULTS

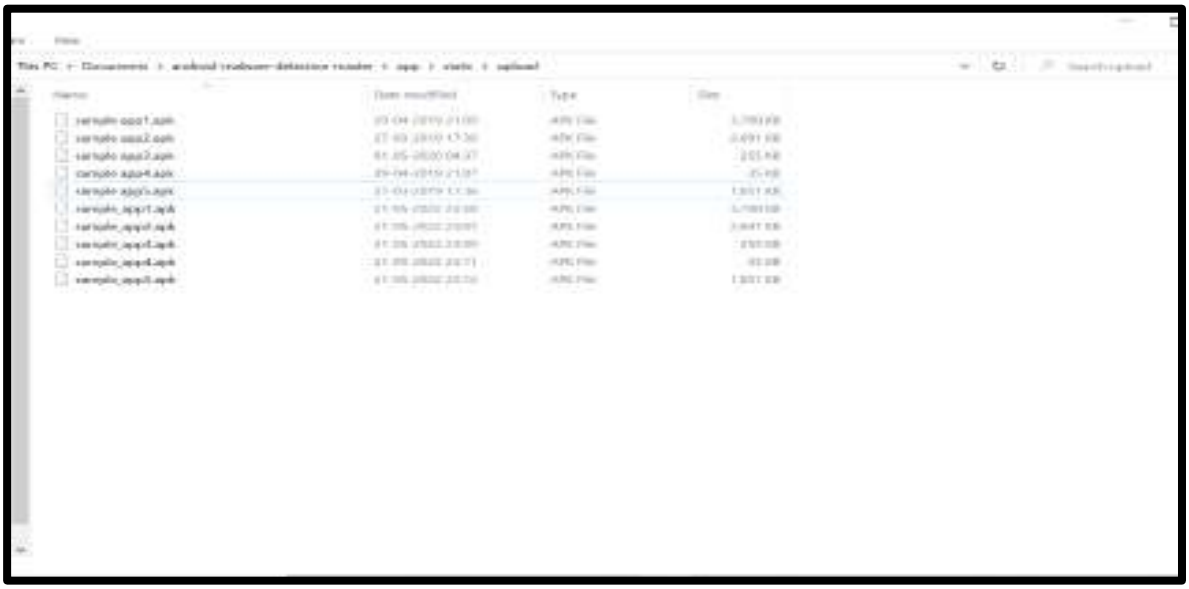
5.1 SCREENSHOTS



Screenshot 5.1.1- Obtaining URL after running the app.y file in project folder.



Screenshot 5.1.2- User interface for uploading sample android malware application APIs

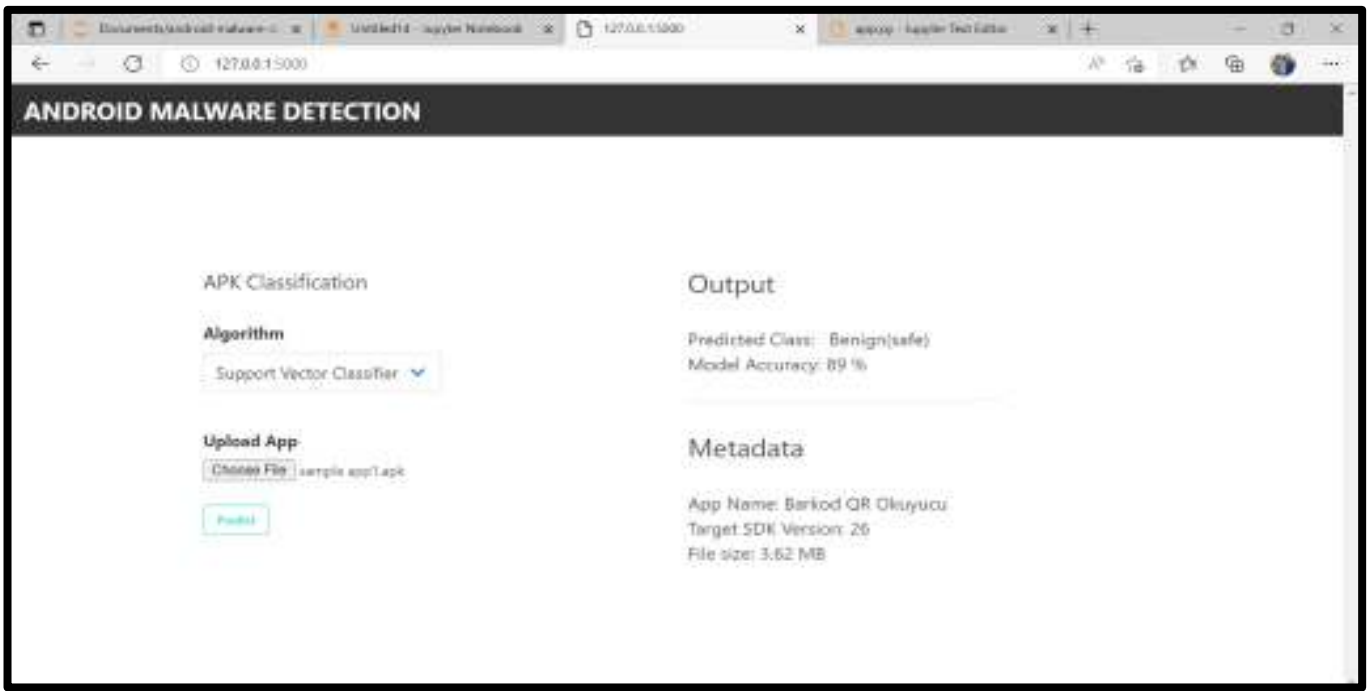


Screenshot 5.1.3- Sample malware application APIs for testing.

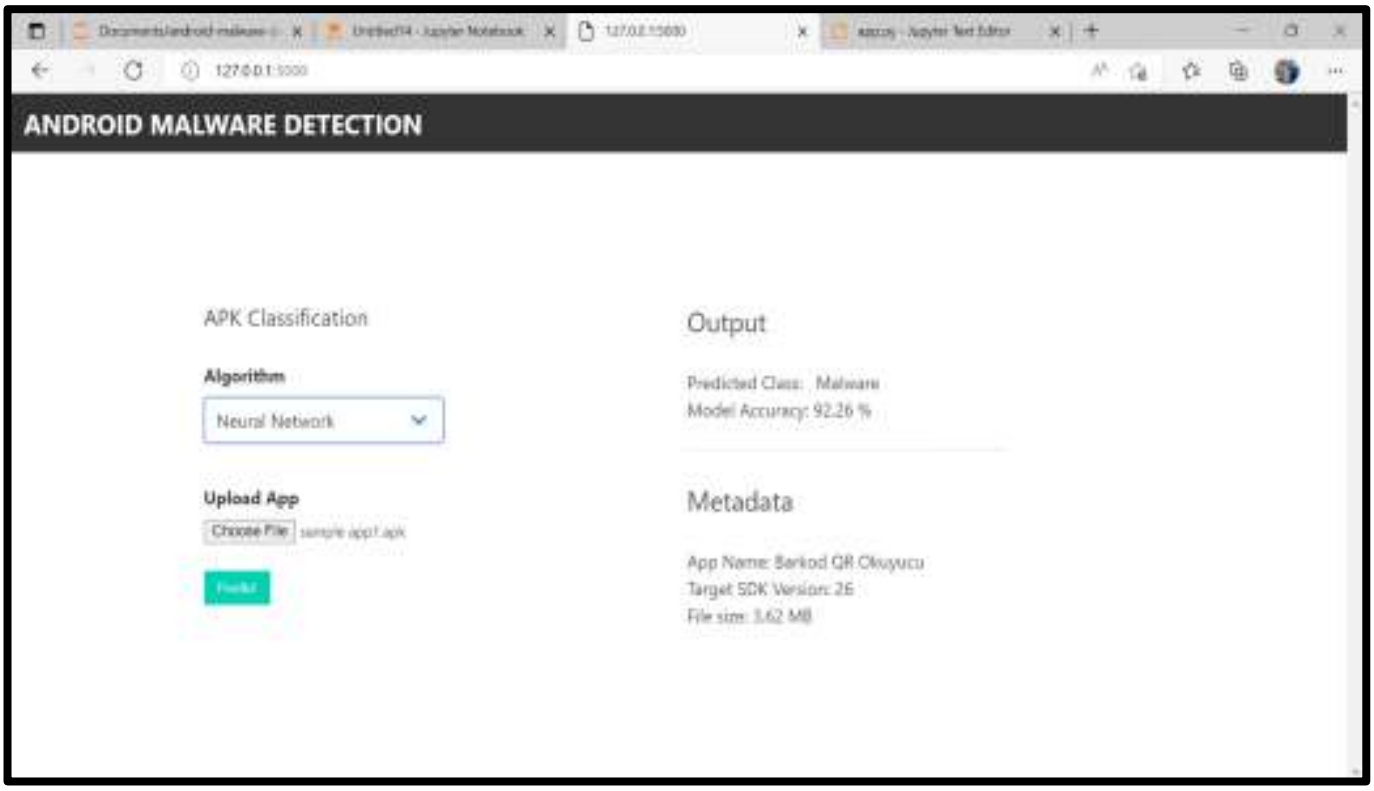
B1 android.permission.BIND_WALLPAPER

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	class	android.p	com.goog	com.moto	com.chan	com.amaz	android.p	de.hafas.a	com.citym	android.p	com.micr	android.p	android.p	android.p	com.sec.e	com.onex	com.ksmo	com.htc.p	com.lge.la
2	benign	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	benign	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	benign	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	benign	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	benign	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	benign	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	benign	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	benign	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	benign	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	benign	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	malware	1	0	0	0	0	1	0	0	1	0	1	0	1	0	0	0	0	1
13	malware	1	0	0	0	0	1	0	0	1	0	1	0	1	0	0	0	0	1
14	malware	1	0	0	0	0	1	0	0	1	0	1	0	1	0	0	0	0	1
15	malware	1	0	0	0	0	1	0	0	1	0	1	0	1	0	0	0	0	1
16	malware	1	0	0	0	0	1	0	0	1	0	1	0	1	0	0	0	0	1
17	malware	1	0	0	0	0	1	0	0	1	0	1	0	1	0	0	0	0	1
18	malware	1	0	0	0	0	1	0	0	1	0	1	0	1	0	0	0	0	1
19	malware	1	0	0	0	0	1	0	0	1	0	1	0	1	0	0	0	0	1
20	malware	1	0	0	0	0	1	0	0	1	0	1	0	1	0	0	0	0	1

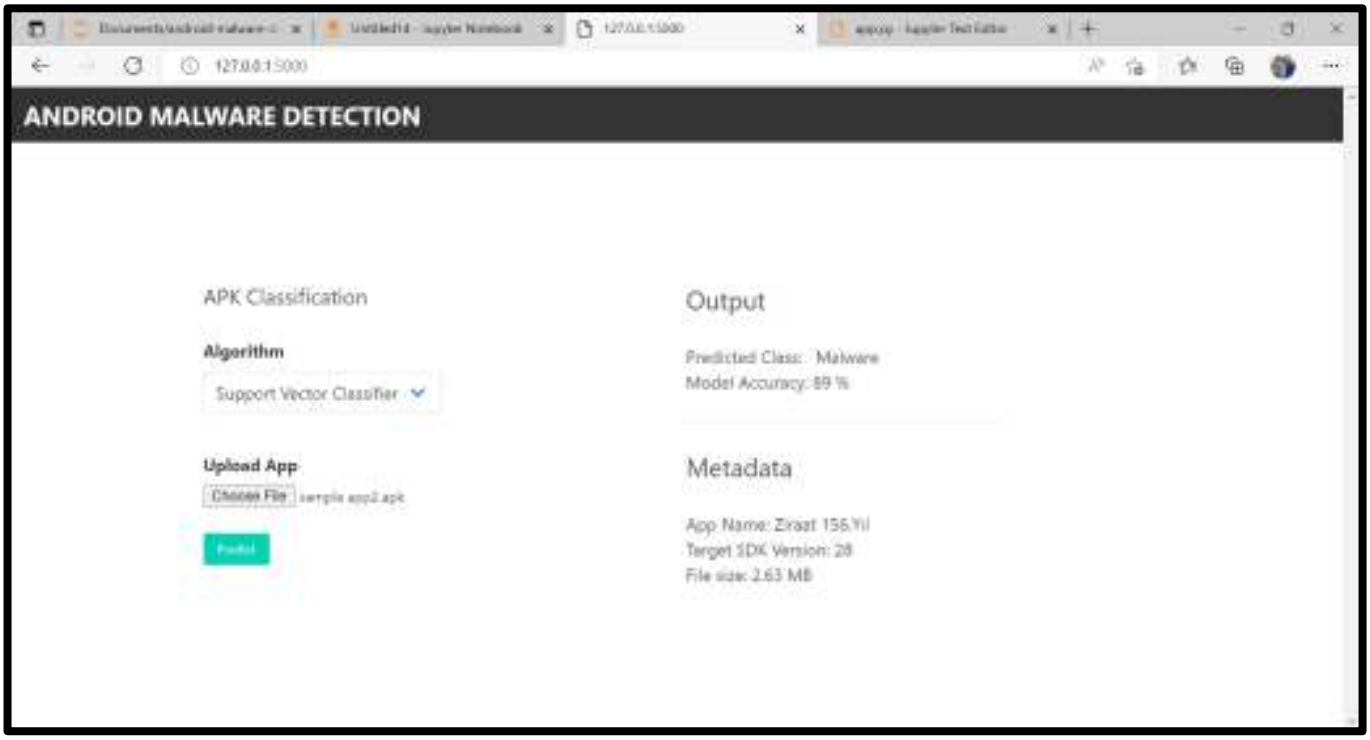
Screenshot 5.1.4- Discriminatory features selected using genetic algorithm



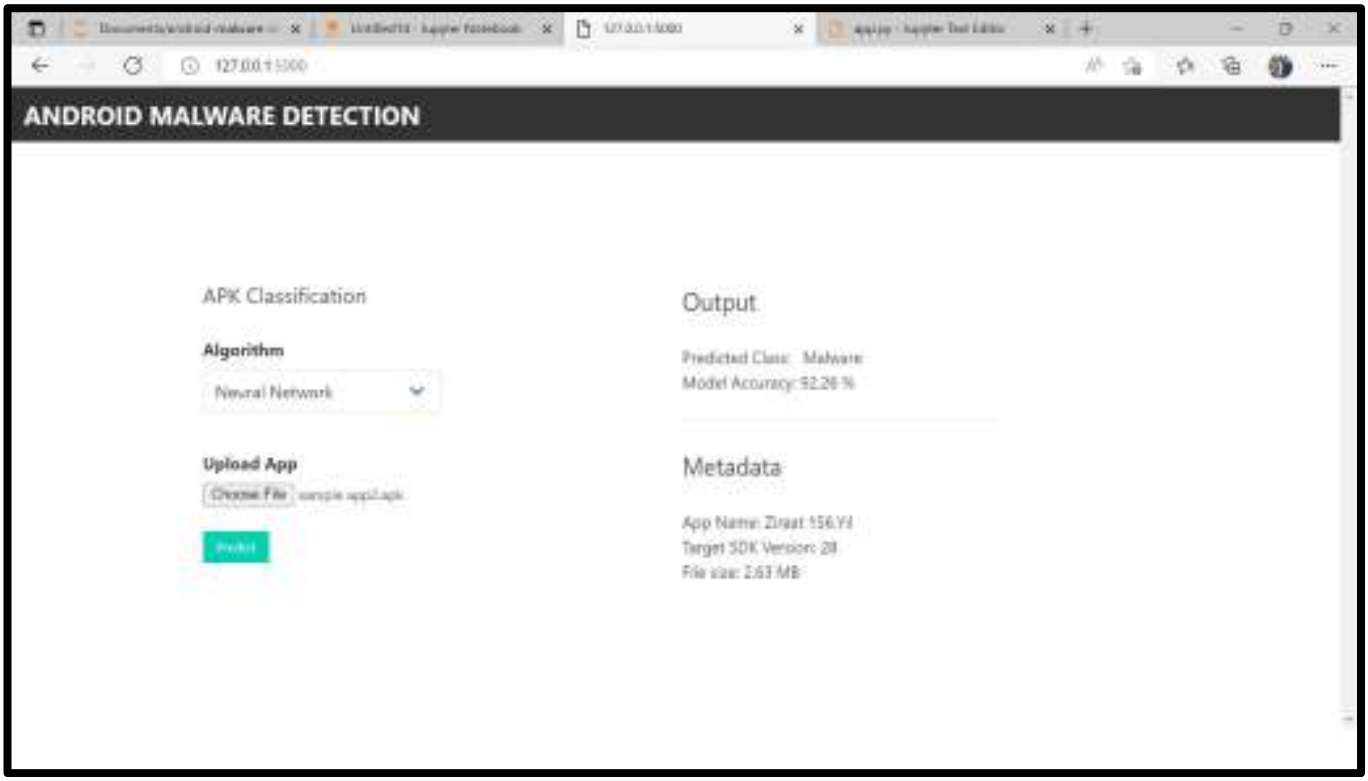
Screenshot 5.1.5- Testing Malware Sample 1 using Support Vector Machine (SVM) Model



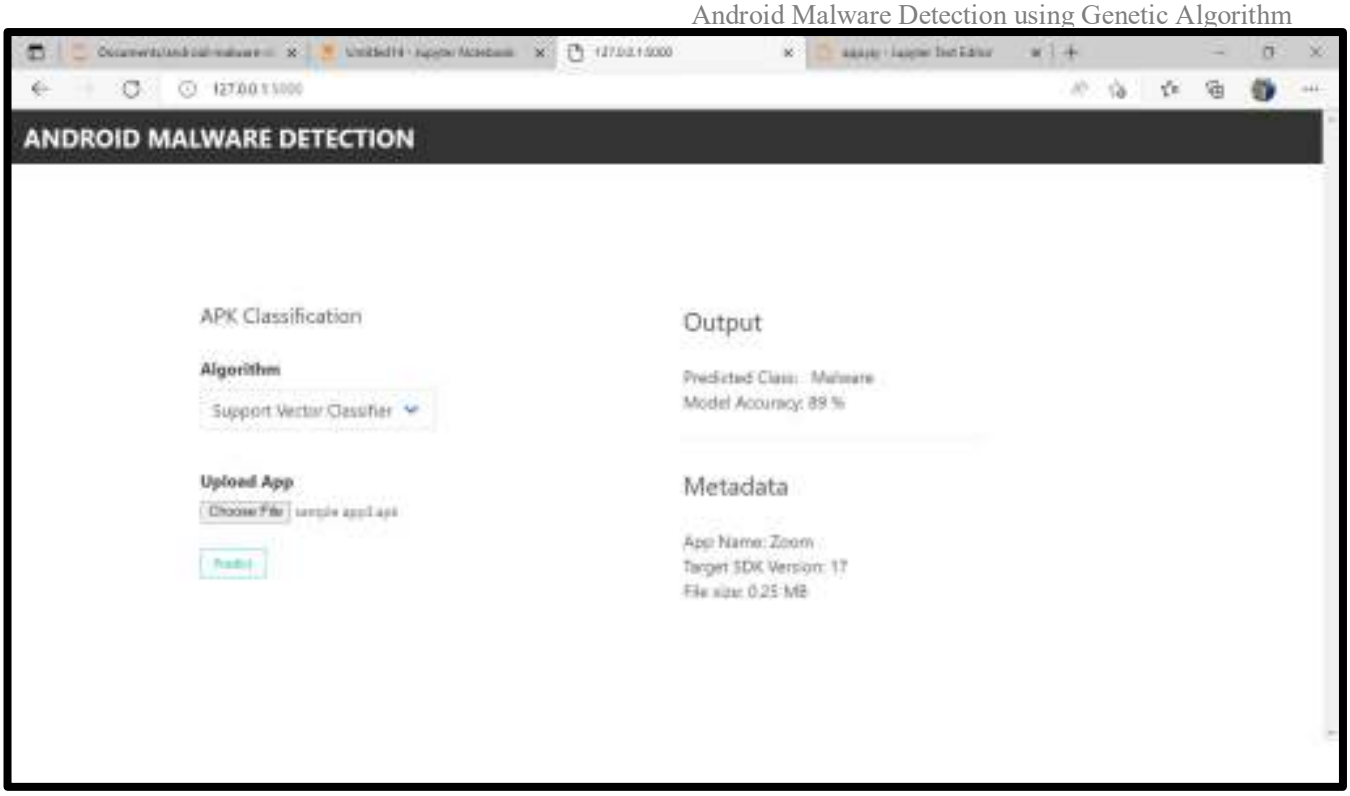
Screenshot 5.1.6- Testing Malware Sample 1 using Neural Network (NN) Model



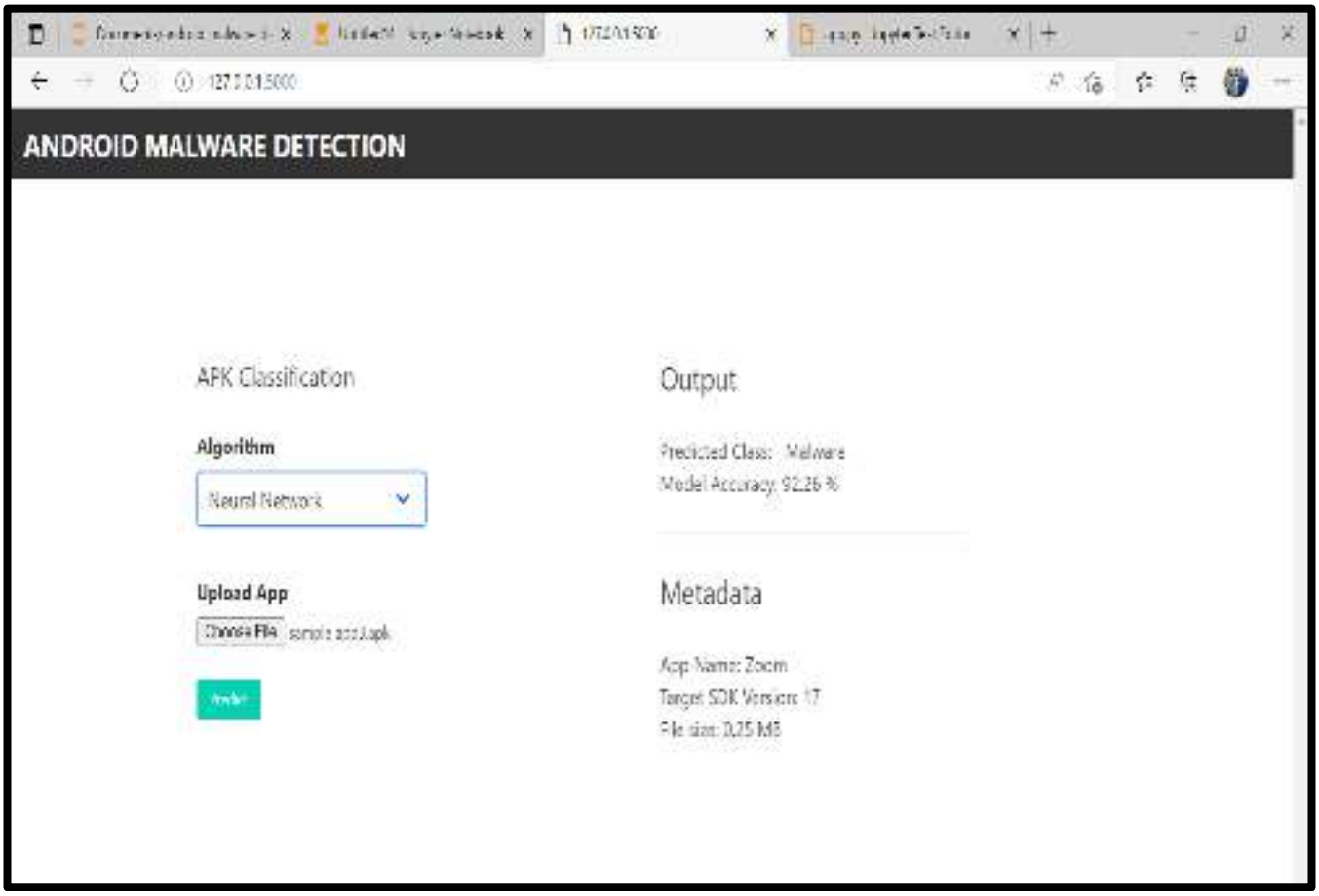
Screenshot 5.1.7- Testing Malware Sample 2 using Support Vector Machine (SVM) Model



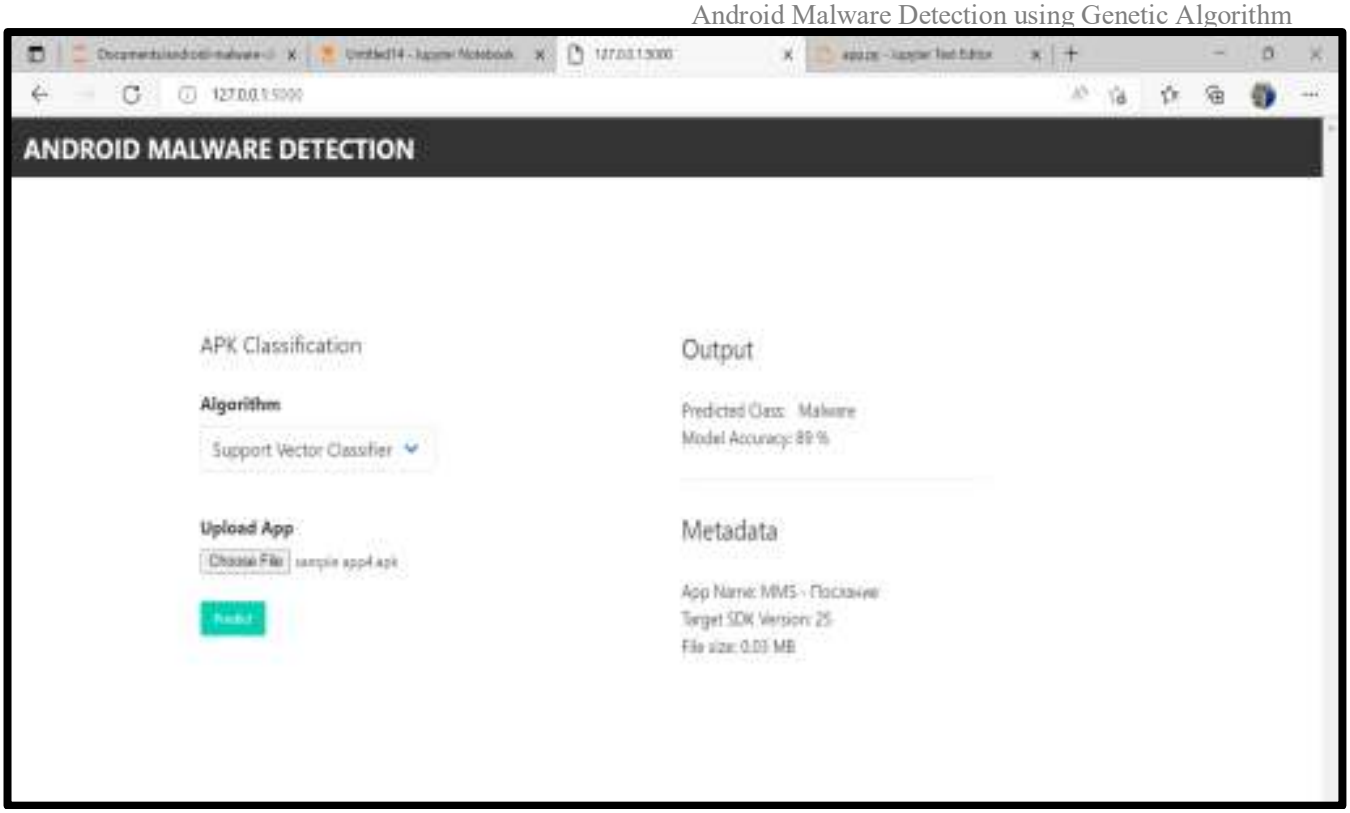
Screenshot 5.1.8- Testing Malware Sample 2 using Neural Network (NN) Model



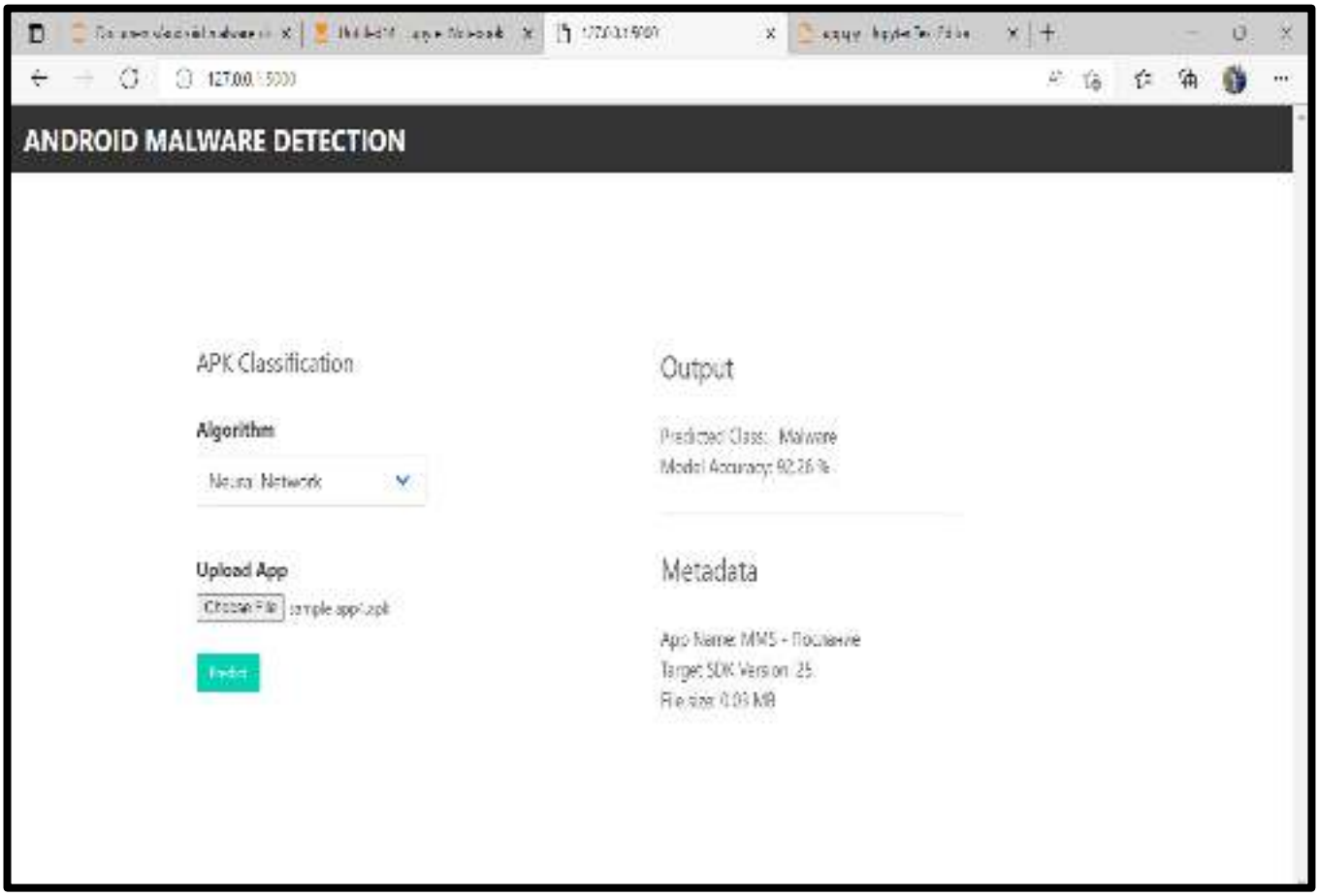
Screenshot 5.1.9- Testing Malware Sample 3 using Support Vector Machine (SVM) Model



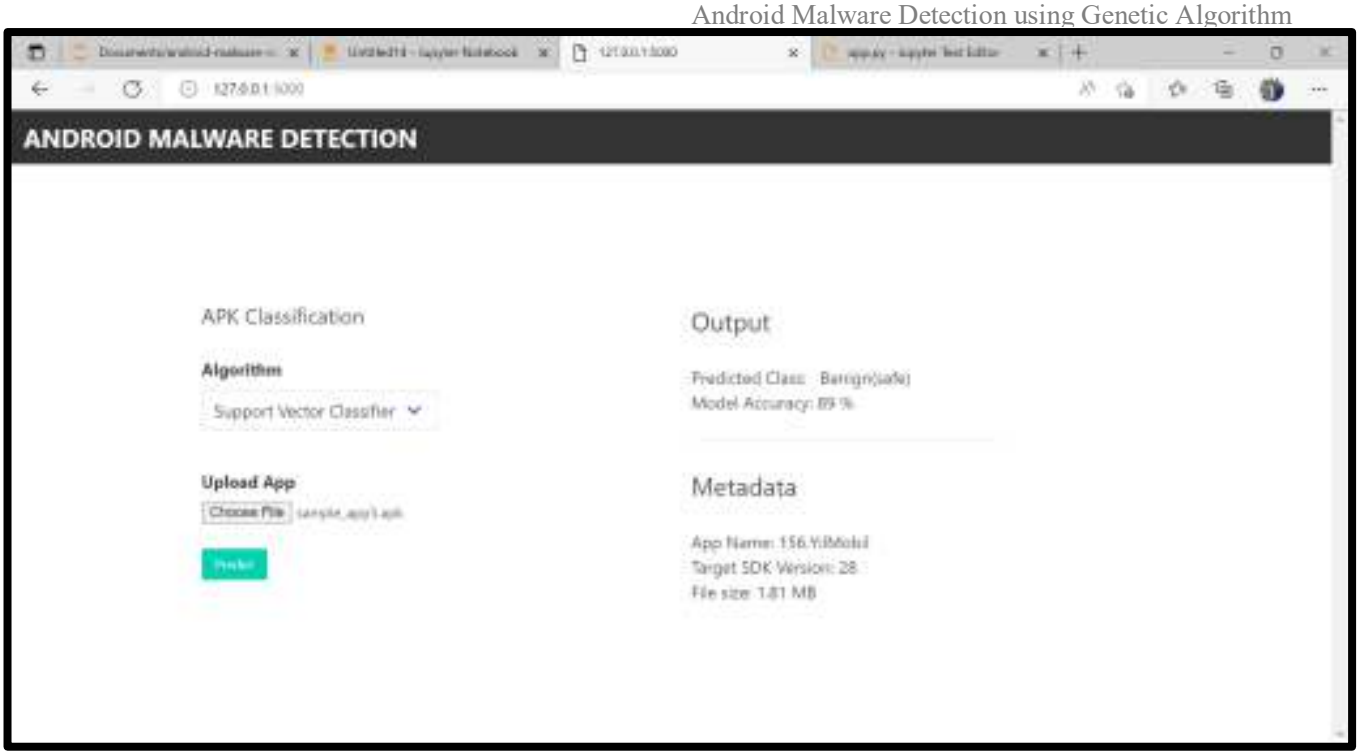
Screenshot 5.1.10- Testing Malware Sample 3 using Neural Network (NN) Model



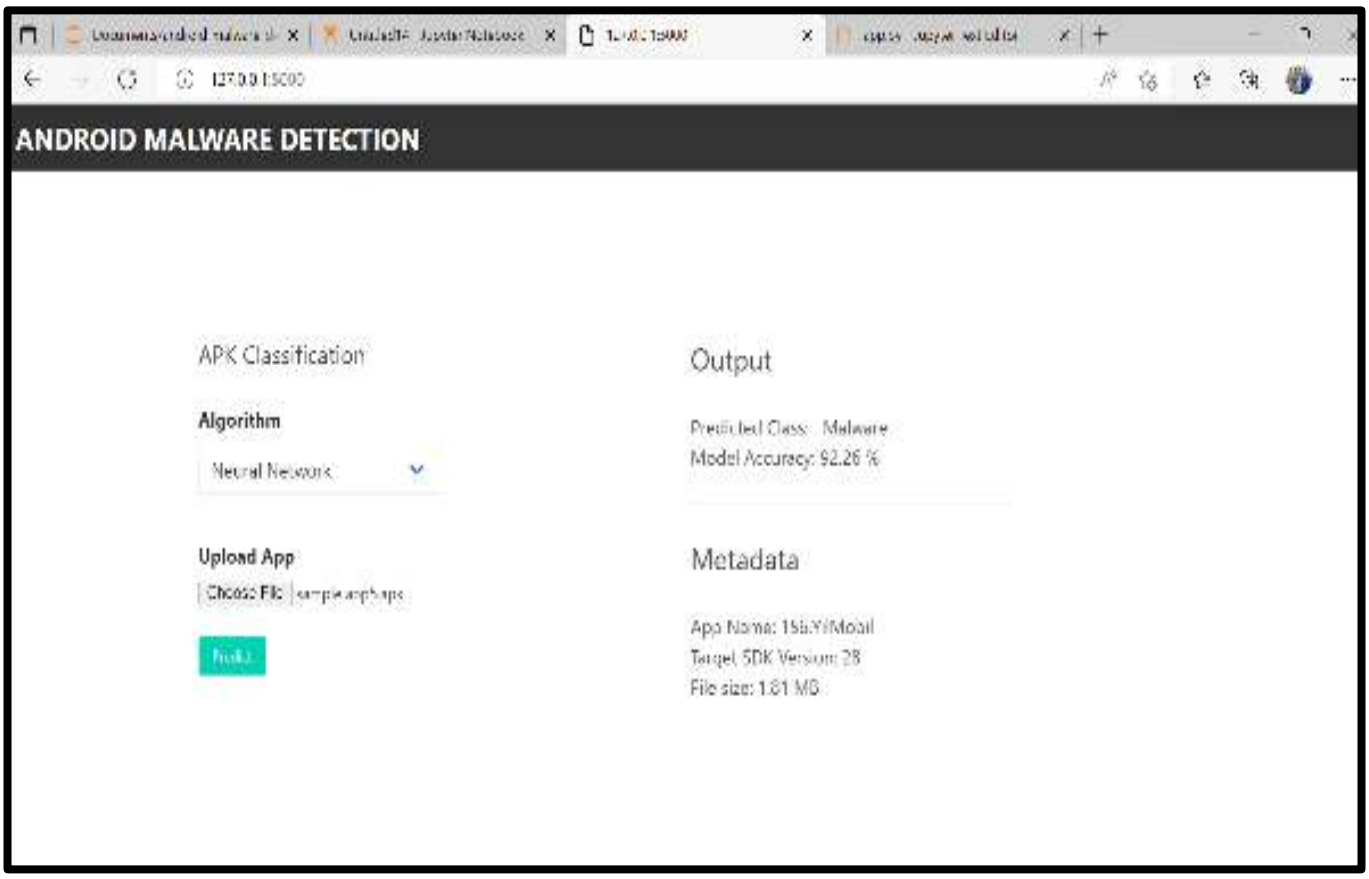
Screenshot 5.1.11- Testing Malware Sample 4 using Support Vector Machine (SVM) Model



Screenshot 5.1.12- Testing Malware Sample 4 using Neural Network (NN) Model



Screenshot 5.1.13- Testing Malware Sample 5 using Support Vector Machine (SVM) Model



Screenshot 5.1.14- Testing Malware Sample 5 using Neural Network (NN) Model

5.2 RESULT ANALYSIS

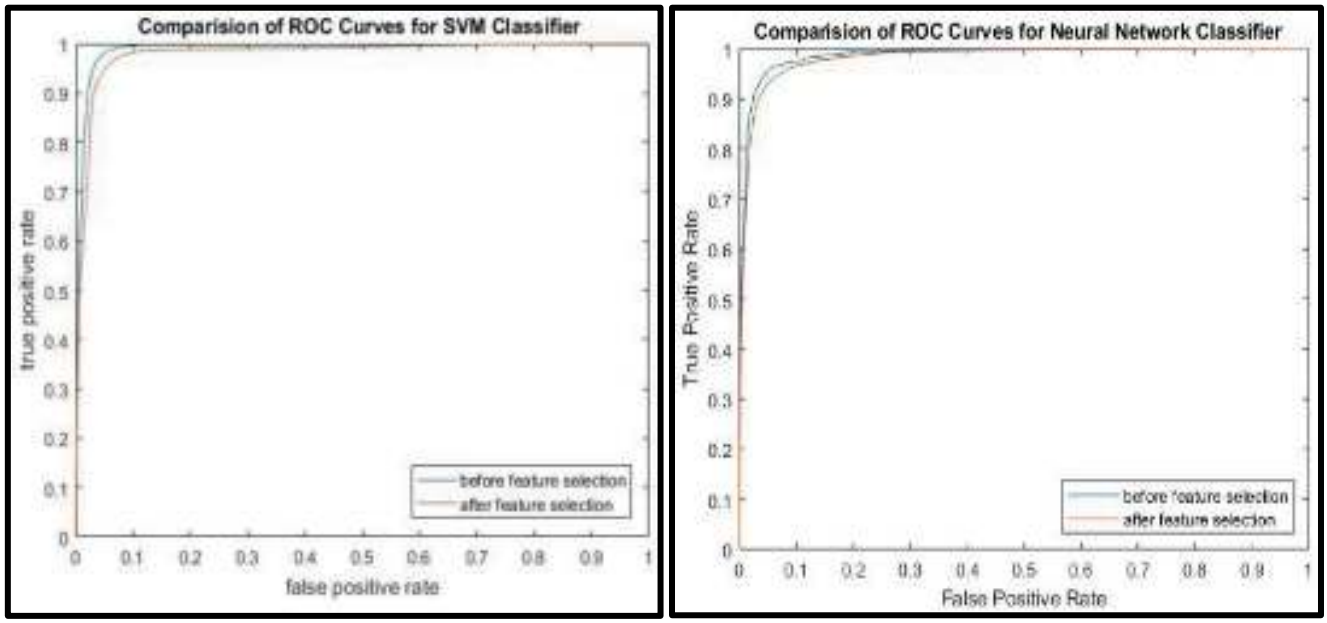
The proposed work has been performed on a dataset of around 40,000 APKs consisting of two categories: 20, 000 Malware or malicious applications and 20,000 Goodware or benign applications. The APKs are reverse engineered to extract features. A CSV is generated consisting of 99 features with class labels as Malware (represented by 0) and Goodware (represented by 1). The primary purpose of the work is selection of optimized feature subset for which Genetic Algorithm has been used. The discriminatory features selected by Genetic Algorithm are fed as input to train Support Vector Machine and Neural Network classifiers. The parameters for Support Vector Machine are set as follows: Radial Basis Function (RBF) as kernel function and number of folds for cross-validation as 10. The number of hidden layers used for the feed-forward Neural Network is one of size 40.

The performance of these two classifiers in distinguishing between Malware and Goodware is compared before and after feature selection.

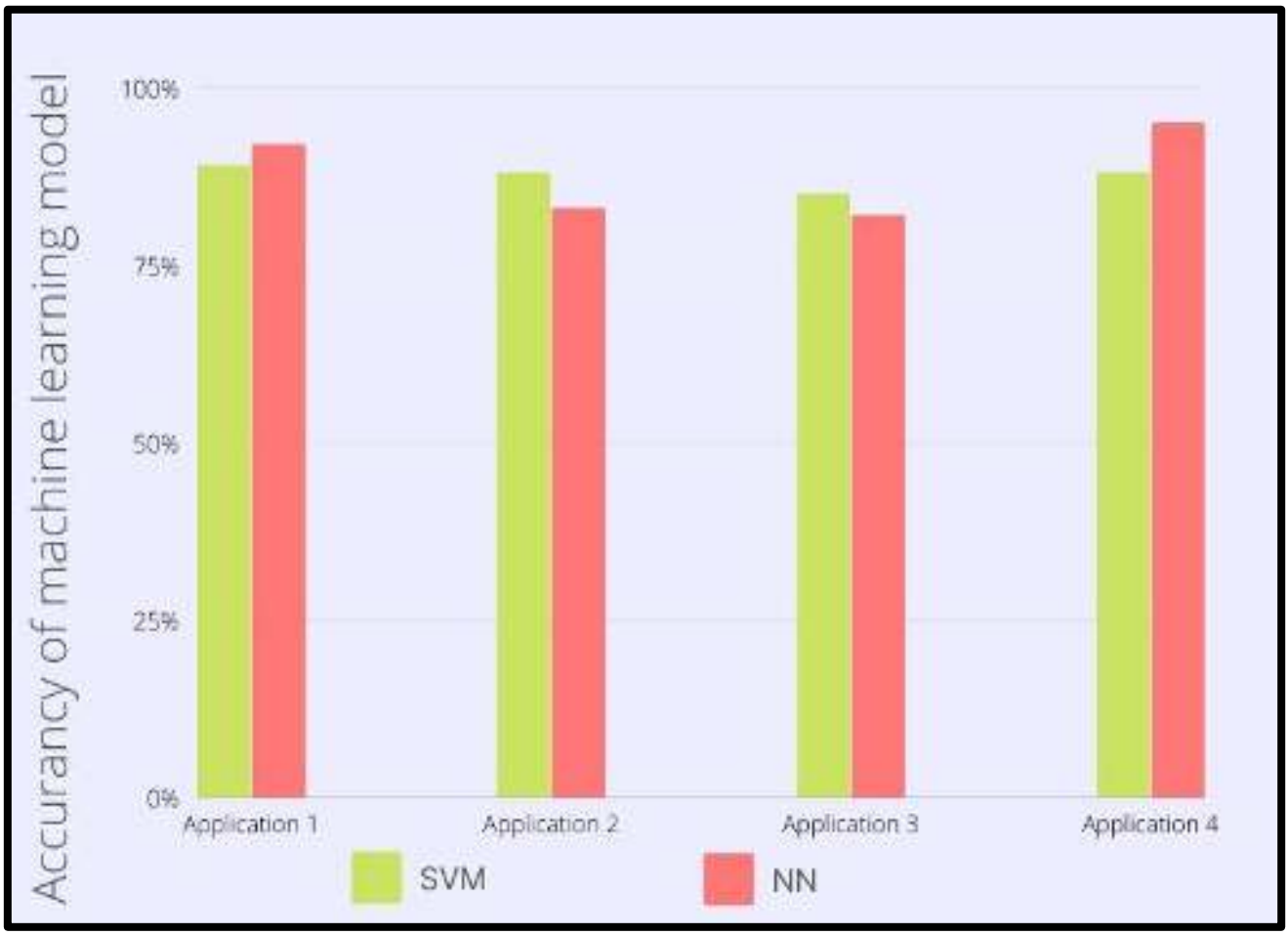
Algorithm	No of features before feature selection	AUC	Features Selected	AUC
Support Vector Machine	99	.9891	33	.9803
Neural Network	99	.9876	40	.9828

Table 5.2.1- Feature selected by genetic algorithm for different classifiers and accuracy obtained with selected features

Table 5.2.1 shows the features selected by the Genetic algorithm for different classifiers and classification accuracy of classifier with the selected subset of features obtained from Genetic algorithm. It can be analyzed from table I that the AUC for both classifiers is preserved to quite an extent with significant reduction in number of features. Below graphs shows the ROC curve for different classifiers before and after feature selection. ROC curves for the Support Vector Machine and Neural Network classifiers are shown in graph 1 and graph 2 respectively. It can be deduced from the ROC curve that classifiers perform well with the selected features.



Graph 5.2.2 and 5.2.3- ROC curves for (1) SVM (2) NN Classifier Before and After Feature Selection



Graph 5.2.4- Comparing Accuracy of SVM and NN machine learning models.

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, subassemblies, 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 to the documented specifications and contains clearly defined inputs and expected results.

6.2.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they 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 centered on the following items:

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

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

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

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

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows, data fields, predefined processes.

6.2.4 TEST CASES

Test case ID	Test case name	Purpose	Input	Output
1	Application test 1	To check if the SVM algorithm performs its task with given datasets	Upload malware APK file	Malware are predicted correctly
2	Application test 2	To check if the NN algorithm performs its task with given datasets	Upload goodware APK file	Goodware are predicted correctly
3	Application test 3	To check if the given file extension other than APK it does not perform	Uploading the file other than the APK extension	Malware/Goodware not predicted correctly

7. CONCLUSION

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 very important to design a framework which can detect such malwares with accurate results. Where signature-based approach fails to detect new variants of malware posing zero-day threats, machine learning based approaches are being used. The proposed methodology attempts to make use of evolutionary Genetic Algorithm to get most optimized feature subset which can be used to train machine learning algorithms in most efficient way. From experimentations, 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

As, the number of dangers presented to Android platforms is growing day to day, spreading primarily via malicious apps or malwares, thus it is extremely essential to develop a framework which can identify such malwares with accurate results. The suggested approach tries to make use of evolving Genetic Algorithm to obtain most optimal feature subset which can be utilized to train machine learning algorithms in most efficient manner. Thus, decreasing the training complexity of the classifiers Further study may be done utilizing bigger datasets for better results and examining the impact on other machine learning methods when used in combination with Genetic Algorithm.

8. BIBILOGRAPHY

8. BIBILOGRAPHY

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.

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

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

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

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

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

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

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

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

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

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

8.2 WEBSITES

https://ieeexplore.ieee.org/document/8769039
https://jespublication.com/upload/2021-V12I1037.pdf

8.3 GITHUB LINK

https://github.com/Sukrutha-101/Project

9. PAPER PUBLICATION

International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 06 Issue: 06 | June - 2022 Impact Factor: 7.185 ISSN: 2582-3930

ANDROID MALWARE DETECTION USING GENETIC ALGORITHM

Najeema Afrin , P Likhitha Krishnaja, Adi Pranay, N Sukrutha

Affiliated to JNTUH, Dept. Of CSE, CMR Technical Campus, Hyderabad, Telangana, India

ABSTRACT

Android has the biggest global market share due to its open-source nature and Google support because it is the most widely used operating system in the world, it has attracted the attention of cyber criminals who use it to spread harmful software applications. This research provides a successful machine -learning based method for malware detection on Android using an evolutionary genetic algorithm feature selection that is discriminatory. The Genetic Algorithm is used to select characteristics before you use machine learning classifiers, make sure you know how good they are at detecting malware. After feature selection, the results are compared. The results of the tests show that genetics is a viable field of study. The algorithm returns the best optimal feature subset, reducing the feature dimension. To a fraction of the original feature set a classification accuracy of greater than 94 percent is considered excellent once feature selection was preserved

Keywords: Android, Genetic algorithm, Malware, feature selection, Classifiers, cyber-criminal.

I. INTRODUCTION

The purpose of this studies is to increase a gadget-studying-primarily based approach for Android malware detection that makes use of a Genetic algorithm to select most advantageous features. On this studies machine gaining knowledge of classifiers are trained the use of selected capabilities from the Genetic set of rules, and their ability to hit upon malware earlier than and after characteristic selection is compared. The effects of the experiments show that the Genetic set of rules offers the pleasant surest function subset, decreasing the characteristic dimension to much less than half of the unique function set. Given the growing number of Android malware variants, an effective malware detection gadget for Android malware is mandatory. In contrary to signature-based totally methods, which want frequent signature database updating, machine studying-based tactics can be employed in mixture with static and dynamic analysis.

II. METHODOLOGY

Genetic algorithm has been employed in the proposed work because of its ability to discover a feature subset picked from the original feature vector that delivers the greatest accuracy for classifiers on which they are trained. It has previously been used in conjunction with machine learning and deep learning algorithms to find the best feature subset. Feature extraction using the Androguard tool and feature selection using the Genetic Algorithm are the two main components of the suggested technique. Finally, for assessment, the selected characteristics are supplied into machine learning algorithms. Static features are derived from AndroidManifest.xml, which provides all of the pertinent information about the Apps required by any Android platform. The Androguard utility was used to disassemble the APKs and extract the data.

III. MODELING AND ANALYSIS

Selecting the most crucial characteristics in malware detection is a vital level since it has a primary affect on the quality of experimental results. Working on a low-dimensional feature vector with simply discriminating traits may also assist lessen the getting to know classifier's computational value. The CSV with all characteristics is put thru the Genetic algorithm, which returns the most excellent subset of functions for the gadget mastering primarily based classifier. The capabilities chosen are represented by using binary forms termed chromosomes, wherein the feature is represented by way of 1 if it's far blanketed and 0 if it's miles omitted within the chromosome. The genetic set of rules continues track of a populace of traits or chromosomes, as well as their health rankings, such that chromosomes with higher health rankings are prioritized.

ARCHITECTURE DESCRIPTION:

Android APKs:

Exclusive pairs of Android apps are available: reverse engineering is used to extract characteristics together with permissions and the remember of App additives including hobby, offerings, and content vendors. Those characteristics are signified as a function vector in Csv record format, with the elegance labels Malware and Goodware displayed by zero and 1 consisting of each.

Feature Vector: As follows, capabilities are retrieved and mapped to a function vector: App additives: A function vector is generated the use of the counts of app additives consisting of hobby, offerings, content providers, and Broadcast Receivers. Permissions: The function dimensional vector space, with a dimension set to one if the app x contains the characteristic and 0 in any other case.

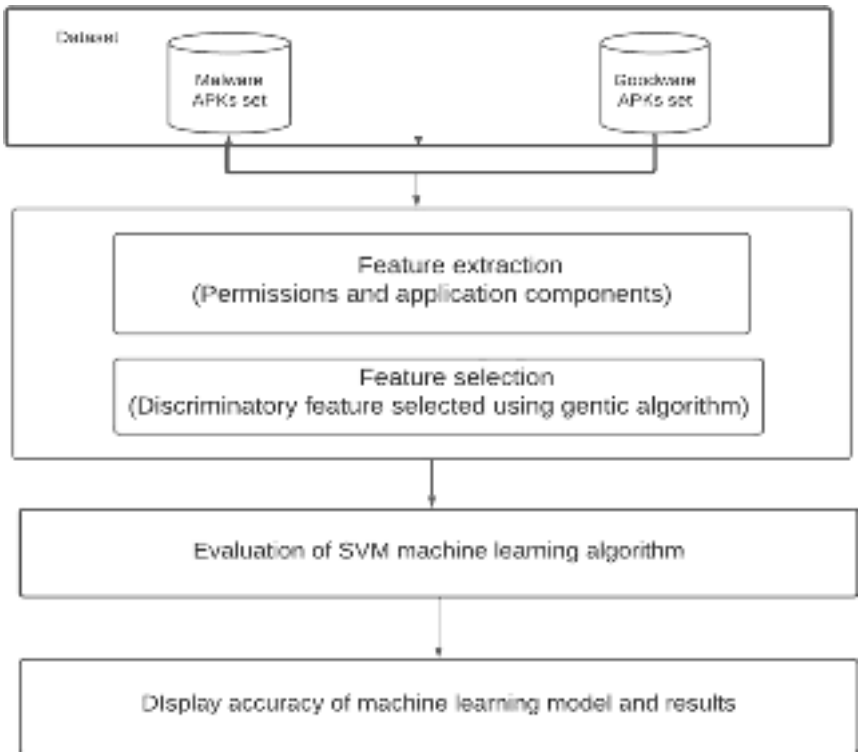


Figure 1: Architecture

IV. RESULTS AND DISCUSSION

The analysis changed into accomplished on a dataset of roughly 40,000 APKs divided into categories: 20,000 Malware (malicious software) and 20,000 Goodware (harmless software program)

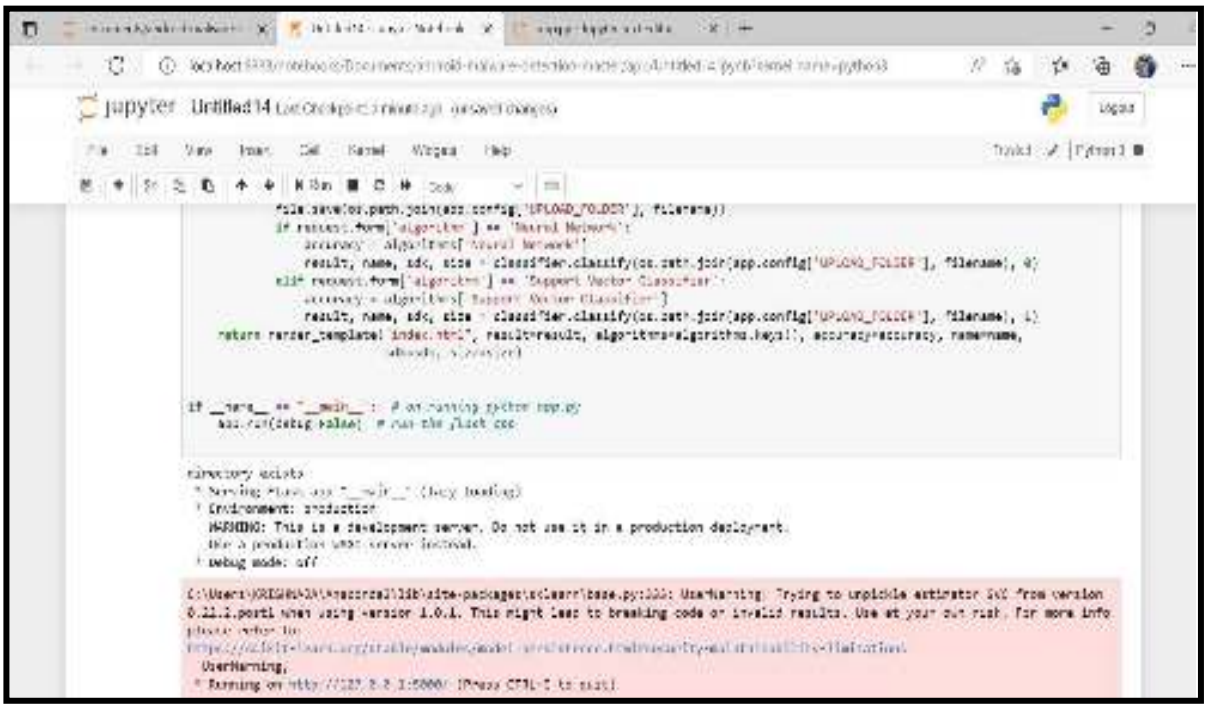


Figure 2: Obtaining URL.

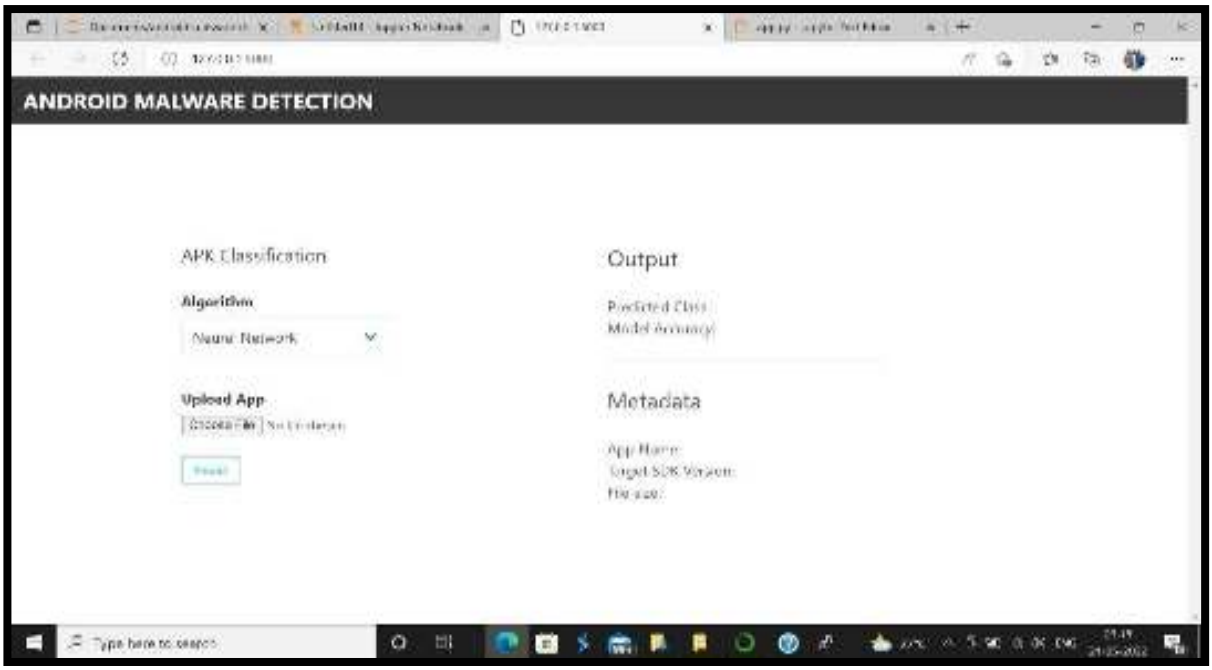


Figure 3 : User Interface.

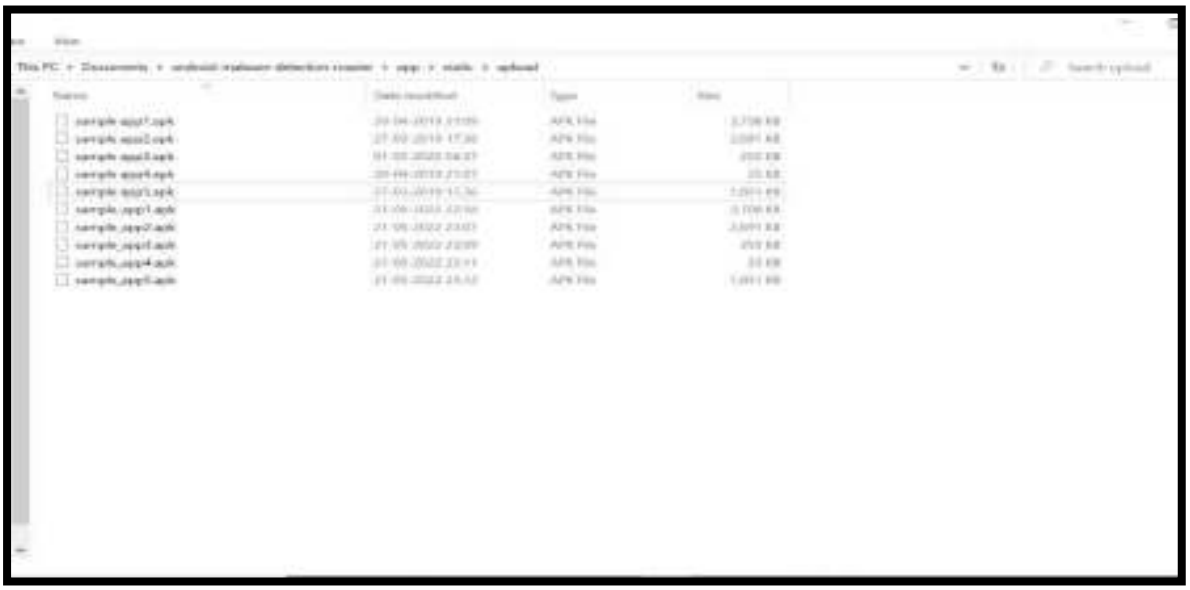


Figure 4 : Sample Malware Applications.

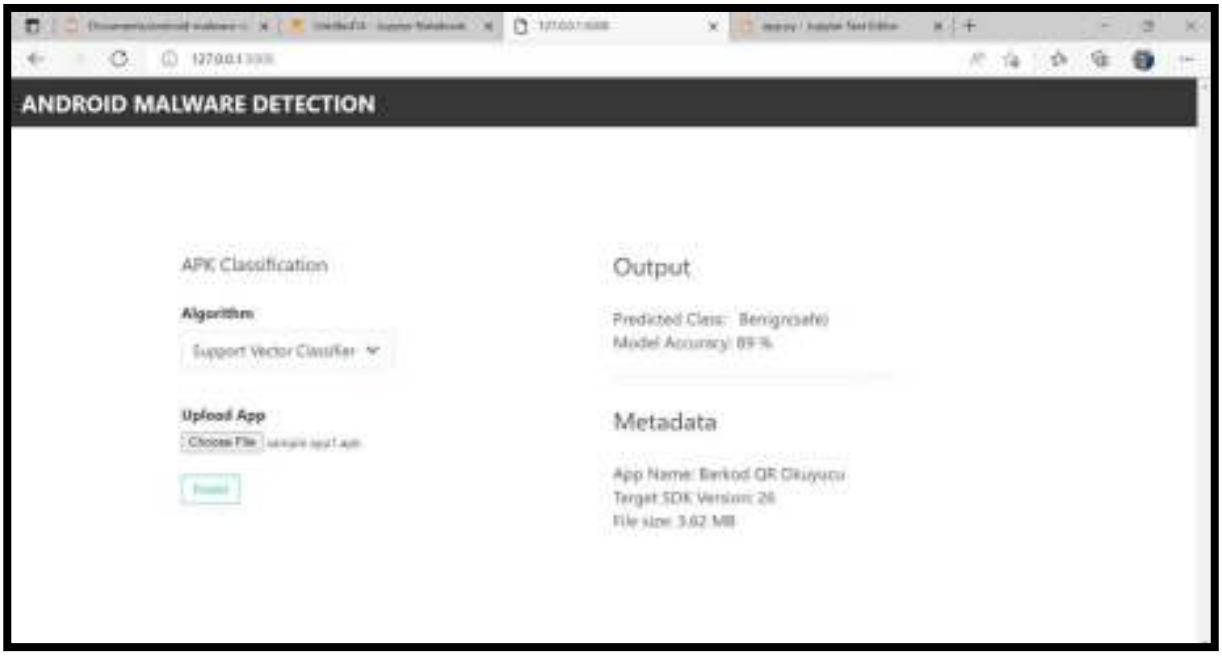


Figure 5 : Testing Malware Sample.

V. CONCLUSION

This section carries all of the vital points.. As the number of dangers posed to Android structures grows every day, spreading frequently thru malicious packages or malware, it's miles essential to increase a framework that could correctly come across such malware. Device learning-primarily based strategies are applied whilst signature-primarily based methods fail to detect new variations of malware posing zero-day risks. The counseled method uses an evolving Genetic algorithm to achieve the best most efficient feature subset that can be utilized to educate device mastering algorithms in the best way

VI. REFERENCES

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

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

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

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

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

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

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

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

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

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

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

[12] A. V. Phan, M. Le Nguyen, and L. T. Bui, "Feature weighting and SVM parameters optimization based

on genetic algorithms for classification problems," Appl. Intell., vol. 46, no. 2, pp. 455–469, 2017.

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

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

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

10. CERTIFICATES

ISSN: 2582-3930

International Journal of Scientific Research in Engineering and Management

is hereby awarding this certificate to

P Likhitha Krishnaja

in recognition the publication of manuscript entitled

ANDROID MALWARE DETECTION USING GENETIC ALGORITHM

published in Ijsrem Journal Volume 06 Issue 06 June 2022

Editor in Chief
E-mail: editor@ijsrem.com

ISSN: 2582-3930

International Journal of Scientific Research in Engineering and Management

is hereby awarding this certificate to

Adi Pranay

in recognition the publication of manuscript entitled

ANDROID MALWARE DETECTION USING GENETIC ALGORITHM

published in Ijsrem Journal Volume 06 Issue 06 June 2022

Editor in Chief
E-mail: editor@ijsrem.com

ISSN: 2582-3930

International Journal of Scientific Research in Engineering and Management

is hereby awarding this certificate to

N Sukrutha

in recognition the publication of manuscript entitled

ANDROID MALWARE DETECTION USING GENETIC ALGORITHM

published in Ijsrem Journal Volume 06 Issue 06 June 2022

www.ijsrem.com

Editor in Chief
E-mail: editor@ijsrem.com

ISSN: 2582-3930

International Journal of Scientific Research in Engineering and Management

is hereby awarding this certificate to

Najeema Afrin

in recognition the publication of manuscript entitled

ANDROID MALWARE DETECTION USING GENETIC ALGORITHM

published in Ijsrem Journal Volume 06 Issue 06 June 2022

Editor in Chief
E-mail: editor@ijsrem.com