An Optimized Technique For Android Malware Detection Using Genetic Algorithm And Machine Learning Model

P Rahul Kiran – 224003087
Cherukuri Rakshith – 224003092
S Vamsi Aakash - 224003133

Guided By
Dr. R RajaKumar

Agenda

- Introduction
- Abstract
- Literature Survey
- Problems Identified
- Objectives
- Proposed System
- Dataset
- Work Plan Carried out
- Requirements
- Sample Output
- References

Introduction

- The increased use of mobile devices in recent years has led to a sharp increase in Android malware threats. Malicious software that attacks on security vulnerabilities in Android devices is known as Android malware.

- Android malware can be causing harm to one's financial status and allow unauthorized access to personal data.

- Since Android malware attacks are becoming more frequent, it's essential to have precisely identifying it

Abstract

- The Android platform has the biggest global market share because of its open source nature and support from Google. Due to the widespread dissemination of malicious programmes, the most widely used operating system in the world has attracted the attention of cybercriminals.
- This project suggests an effective machine-learning-based method for Android malware detection that selects features based on discrimination and uses an evolutionary genetic algorithm.
- Machine learning classifiers are trained using specific features using genetic algorithms, and their ability to malware both before and after feature selection is compared.
- In order to train the model, the genetic algorithm will choose key features from the dataset and exclude irrelevant features. The training model will be built faster and the dataset size will be lowered as a result of this approach.

Literature Survey

S.No	Journal Type , Year	Authors	Title	Methodology	Limitations
1.	IEEE Journal Article,2023	Abhinandan Basik, Jyothi Prakash Singh	Android Malware Detection by Correlated Real Permission Couples Using FP Growth Algorithm and Neural Networks	The proposed approach utilizes the Frequent Pattern (FP) Growth algorithm to identify common pairs of real permissions from Android disassembled code. These pairs are then inputted into a multi-layered neural network model for binary classification, determining if an APK is malware or benignware.	Narrow focus on permission features neglects other potential indicators of malware. Time-consuming and labor-intensive process of reverse engineering.

S.No	Journal Type, Year	Authors	Title	Methodolgy	Limitations
2.	IEEE Journal Article,2021	Carlos Cilleruelo, Enrique-Larriba, Luis De-Marcos, Jose Javier Martinez-Herraiz	Malware Detection Inside App Stores Based on Lifespan Measurements	The paper introduces a machine learning solution for detecting Potentially Harmful Apps (PHAs) in the Google Play Store. By analyzing app lifespans with a unique dataset, it achieves 90% accuracy. While not as accurate as some models, it offers a distinct approach for PHA detection, suitable for integration with other methods.	The limitations are the accuracy of the approach, potential bias in PHA identification based on app lifespan, and the inability to detect more sophisticated PHAs.

S.No	Journal Type, Year	Authors	Title	Methodology	Limitations
3.	IEEE Journal Article,2021	Long Nguyen Vu, Souhwan Jung	AdMat: A CNN-on-Matrix Approach to Android Malware Detection and Classification	The study introduces AdMat, a framework that treats Android applications as images and uses Convolutional Neural Networks (CNNs) to differentiate between benign and malicious apps, as well as identify malware families. AdMat achieves an average detection rate of 98.26% across different malware datasets and successfully recognizes over 97.00% of different malware families with limited training data.	The limitations of AdMat include vulnerability to dynamic code loading attacks, a trade-off between feature count and detection performance, and concerns about false positives due to similar API names across different class names.

S.No	Journal Type, Year	Authors	Title	Methodology	Limitations
4.	IEEE Journal Article,2022	Beenish Urooj, Munam Ali Shah, Carsten Maple, Md Kamran Abbasi, Sidra Riasat	Malware Detection: A Framework for Reverse Engineered Android Applications Through Machine Learning Algorithms	The paper introduces a machine learning-based method to detect Android malware, achieving 96.24% accuracy with a 0.3% false positive rate. It combines innovative static features, large datasets, and ensemble learning algorithms like AdaBoost and SVM. The model incorporates various features extracted from reverse-engineered Android applications, including permissions, intents, and API calls.	The paper focuses on binary classification using static features from the Android manifest file. Key limitations include: Lack of dynamic feature consideration. Sustainability challenges for future updates. Processing constraints impacting accuracy.

Problems Identified

- Android applications possess numerous features, leading to high-dimensional feature spaces.
- Selecting the features is essential to improve classification accuracy while minimizing computation
- Traditional feature selection methods may struggle to handle zero-day attacks, as they exploit vulnerabilities that are unknown to security experts, making them difficult to detect.

Objectives

- To improve the accuracy of detecting malware by making optimized feature selection.

- To analyse the outcomes of machine learning models with using genetic algorithm and without using genetic algorithm.

Proposed System:

- In this project we are using two machine learning algorithms such as SVM(Support Vector Machine) and ANN(Artificial Neutral Networks).

- App will contains more than 200 features and machine learning will take more time to build model, so we need to optimize features, for that we are 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.

Dataset

- Malgenome Dataset
- It is taken from the Figshare platform
- It contains 215 features, 3799 samples in which 1260-Malware-S, 2539-Benign-B
- Features are of 3 types
- **API Call Signatures**: These represent interactions with system libraries and functions, indicating specific actions or operations performed by the application.
- **Manifest Permissions**: These permissions define what actions or resources an application can access on the device. They are crucial for ensuring security and privacy.
- **Intents**: Intents are messages that allow components to request actions from other components within the application or across different applications. They facilitate communication between different parts of an Android application or between different Android applications.
- Link:

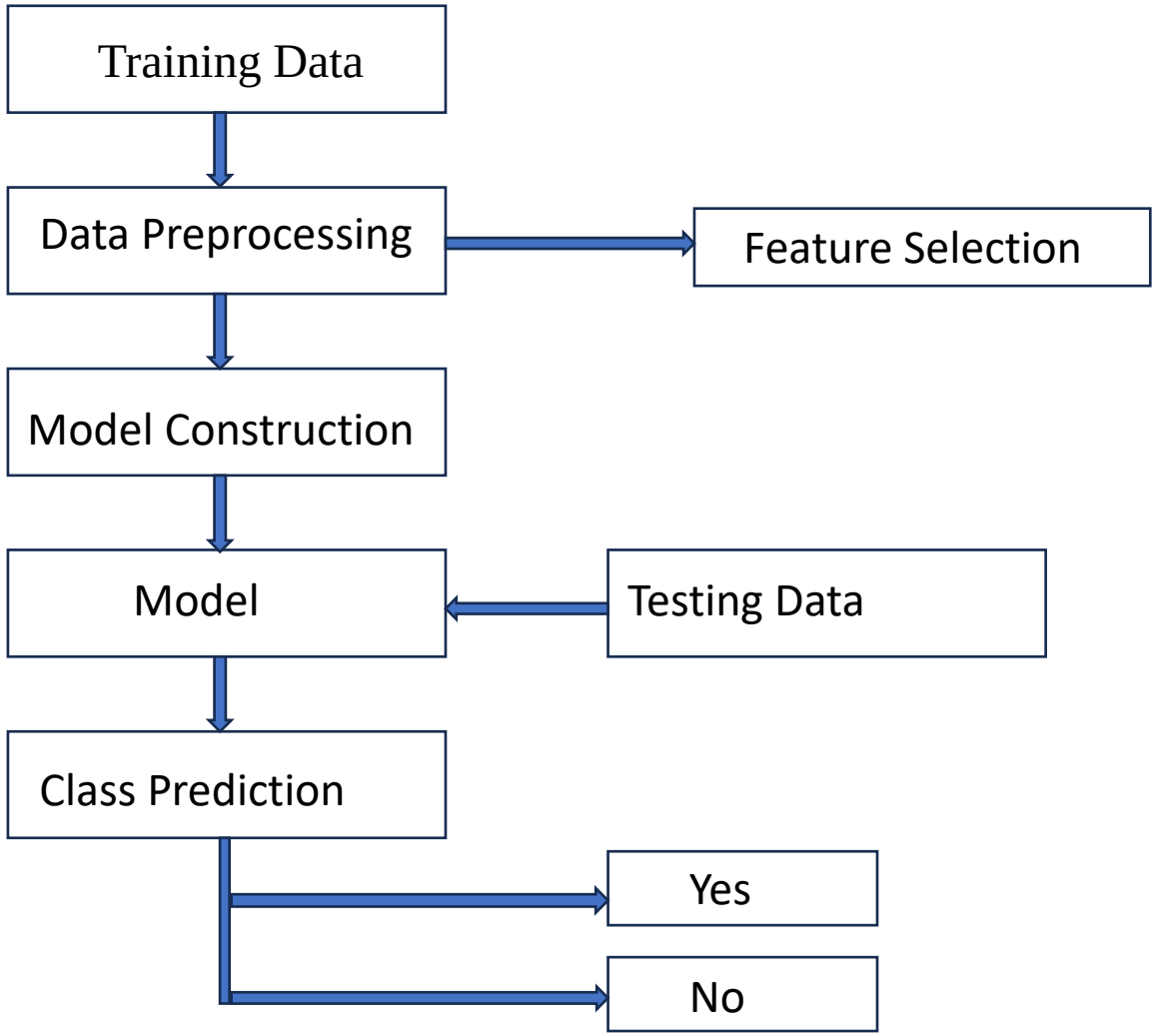
https://figshare.com/articles/dataset/Android_malware_dataset_for_machine_learning_1/5854590

Features:

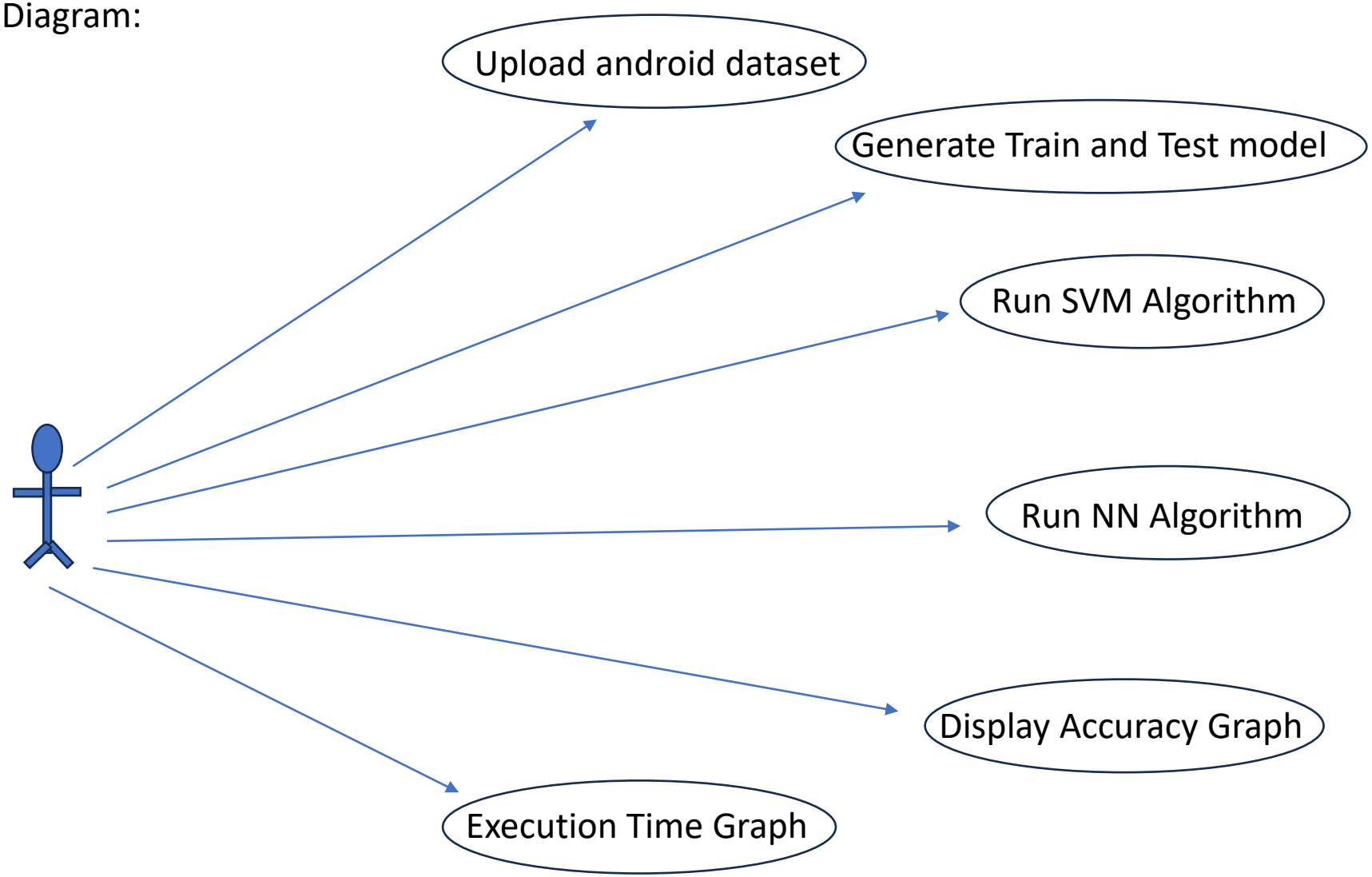
Some of the features in Dataset are

- READ_SMS/WRITE_SMS – Manifest Permission
- READ CONTACTS/WRITE_CONTACTS – Manifest Permission
- READ_EXTERNAL_STORAGE/WRITE_EXTERNAL_STORAGE - Manifest
- ACCESS_FINE_LOCATION – Manifest Permission
- RECORD_AUDIO – Manifest Permission
- CAMERA – Manifest Permission
- INTERNET – Manifest Permission
- Ljava.lang.reflect – API Call Signature
- Javax.crypto.Classes – API call Signature
- android.intent.action.SEND – Intent
- android.intent.action.CALL – Intent
- features

Work Plan


Training Data
Data Preprocessing
Feature Selection
Model Construction
Model
Testing Data
Class Prediction
Yes
No

Use-Case Diagram:



Work Plan Carried Out

- Training Data
- Data Preprocessing
- Feature Selection
- Model Construction
- Model
  - *SVM
  - *SVM with Genetic
- Testing Data
- Prediction

SVM:

- **Optimal Hyperplane:** SVM identifies the optimal hyperplane that maximizes the margin between classes.
- **Support Vectors:** It determines support vectors, which are data points closest to the decision boundary.
- **Parameter Optimization:** Through optimization techniques and regularization parameter tuning, SVM minimizes classification errors while balancing margin maximization. If needed, it utilizes kernel functions to handle non-linearly separable data.
- SVM's decision boundary is determined by the subset of training data points called support vectors, making it memory-efficient for large datasets.

Genetic Algorithm:

- **Optimization Technique**: GA is a metaheuristic optimization algorithm inspired by the process of natural selection and evolution. It uses principles such as selection, crossover, and mutation to iteratively evolve a population of candidate solutions towards an optimal or near-optimal solution.

- **Population-Based Approach**: GA operates on a population of potential solutions, where each solution is represented as a chromosome in the form of a string of genes. Through successive generations, the algorithm evolves better solutions by applying genetic operators like crossover and mutation.

- **Versatility**: GA is versatile and can be applied to various optimization problems, including combinatorial optimization, function optimization, and machine learning tasks, where traditional optimization methods may struggle.

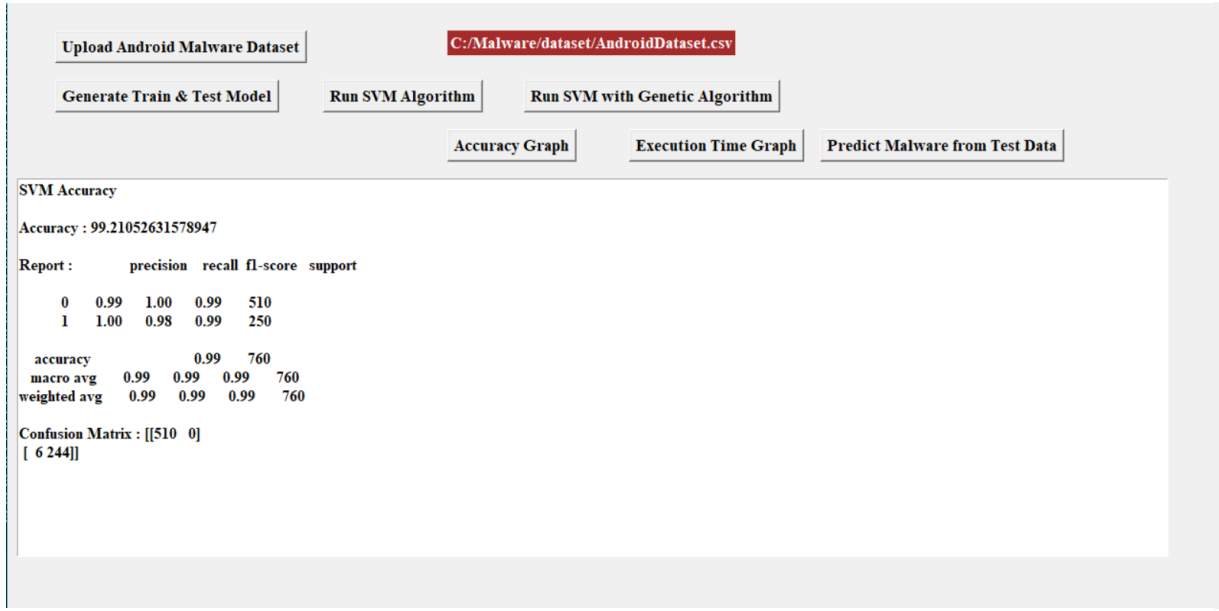
Software Requirements:

Programming Language: Python 3.7 and above
GUI: tkinter package
Packages: Tensorflow, NumPy, Pandas, Matplotlib, Scikit-learn

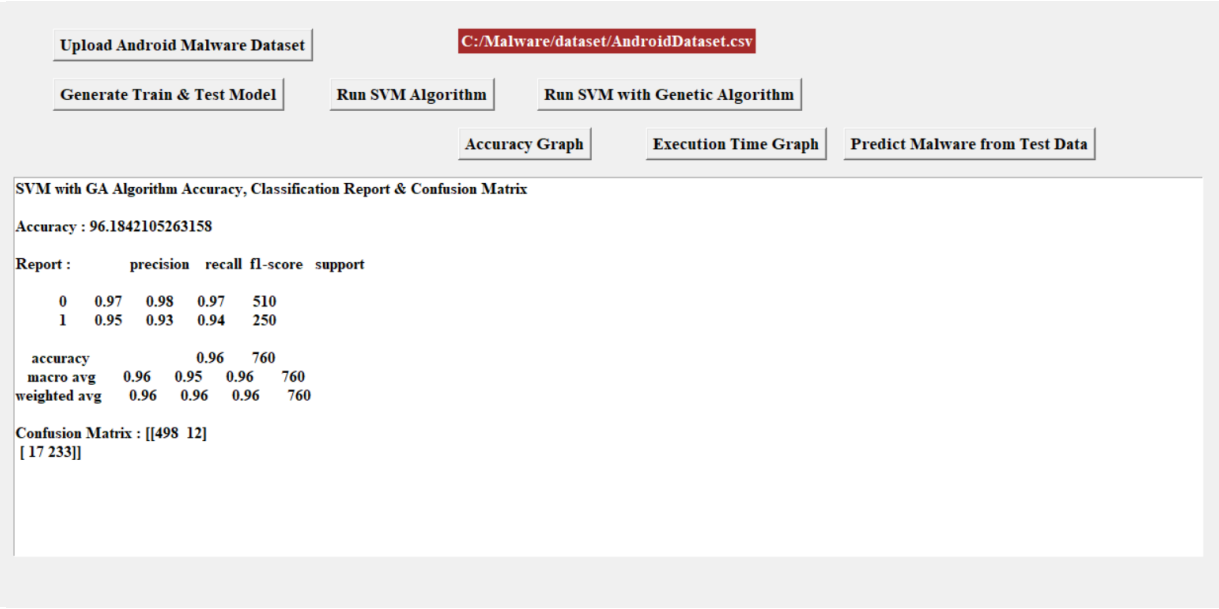
Hardware Requirements:

RAM: 4GB
Hard Disk Drive:128GB

Sample Output:



SVM



SVM with Genetic

References:

- 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.

- https://www.javatpoint.com/genetic-algorithm-in-machine-learning

- https://www.analyticsvidhya.com/blog/2021/06/genetic-algorithms-and-its-use-cases-in-machine-learning/

THANK YOU