

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.

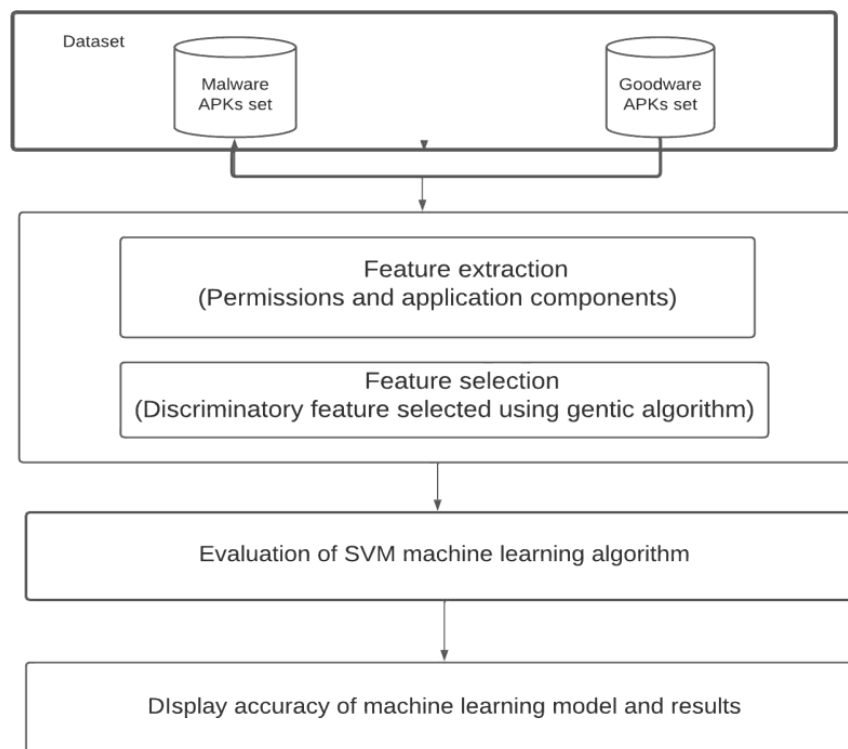
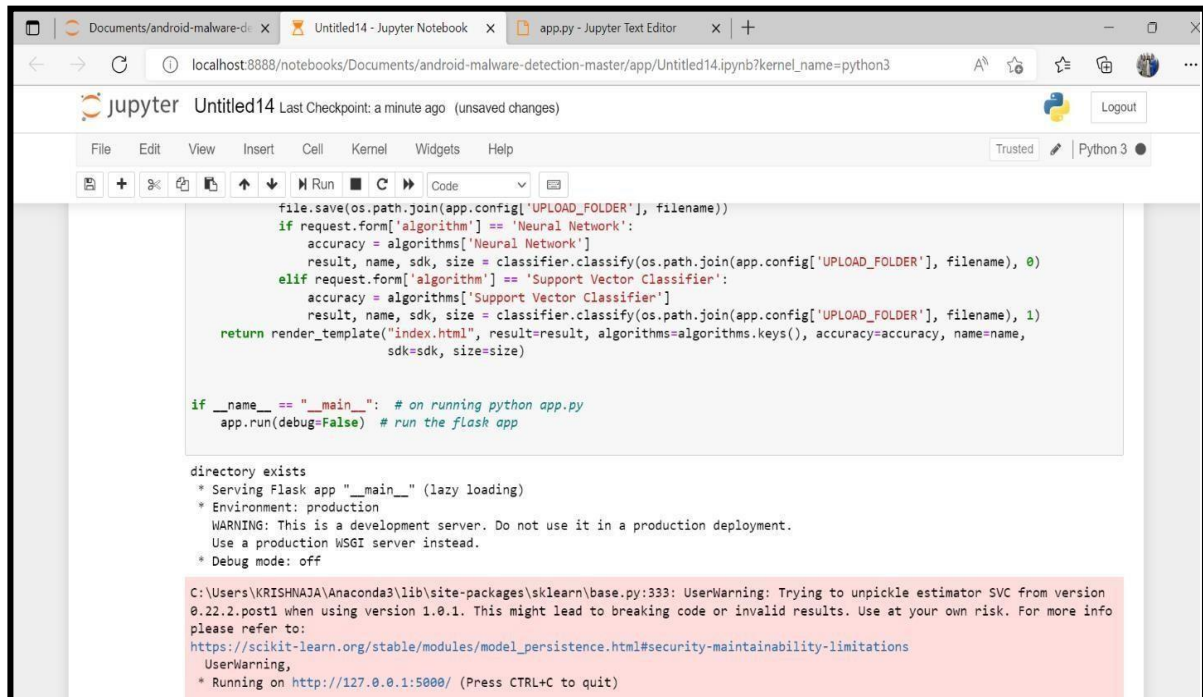


Figure 1: Architecture

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)



```
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

directory exists
* Serving Flask app "__main__" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off

C:\Users\KRISHNAJA\Anaconda3\lib\site-packages\sklearn\base.py:333: UserWarning: Trying to unpickle estimator SVC from version
0.22.2.post1 when using version 1.0.1. This might lead to breaking code or invalid results. Use at your own risk. For more info
please refer to:
https://scikit-learn.org/stable/modules/model_persistence.html#security-maintainability-limitations
UserWarning,
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Figure 2: Obtaining URL.

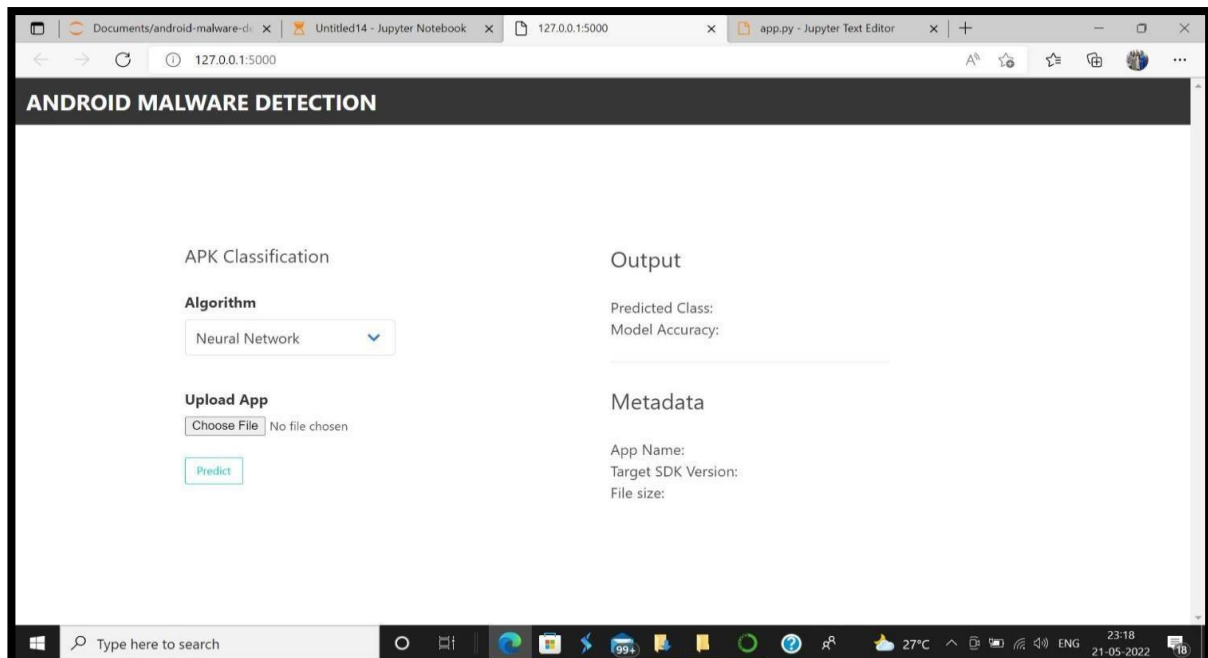


Figure 3 : User Interface.



Figure 4 : Sample Malware Applications.

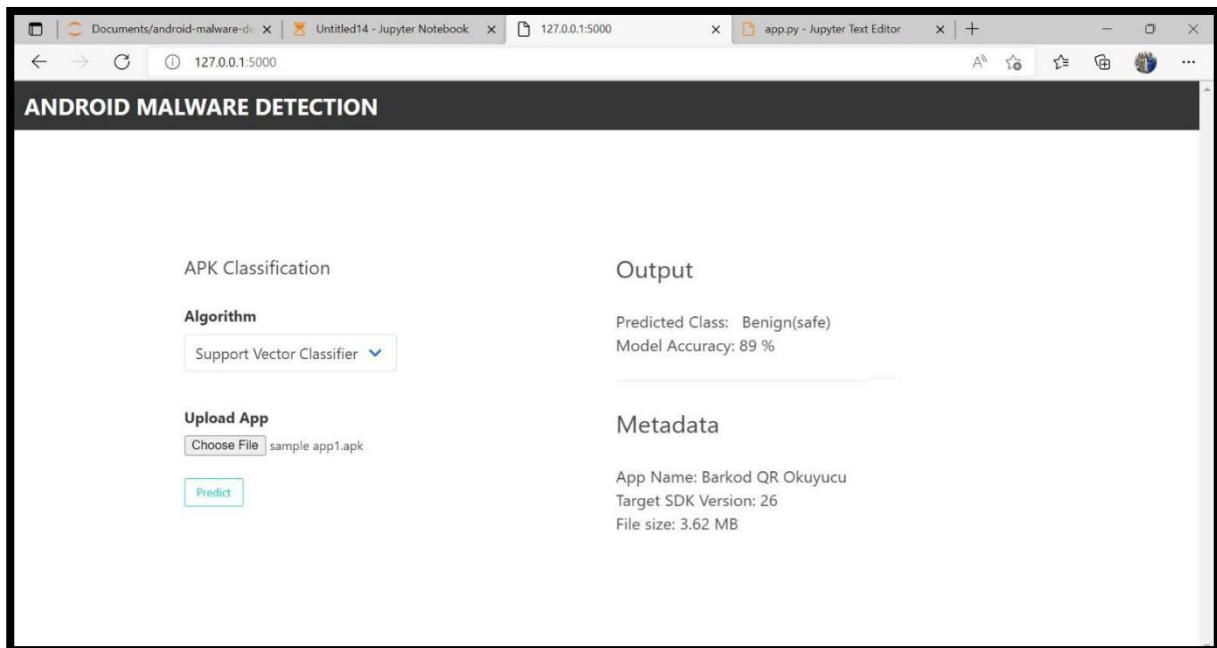


Figure 5 : Testing Malware Sample.

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.