A New Machine Learning Method for Identifying Android Malware by Using Features That Co-Exist

Mrs. Pulagouni Priyanka[1] M.Tech., Assistant Professor.
P. Eduru Basha[2], S. Sai Deepak[3], D. Srija[4], T. Naveen[5]
Computer Science and Engineering & Business Systems Department,
Rajeev Gandhi Memorial College of Engineering and Technology, Nandyal.
[1]Assistant Professor Department of CSE&BS, RGMCET
[2]Department of CSE&BS, RGMCET
[3]Department of CSE&BS, RGMCET
[4]Department of CSE&BS, RGMCET
[5]Department of CSE&BS, RGMCET

Abstract: The mission investigates Android malware detection utilising various datasets, along with Drebin, Malgenome, and CIC_MALDROID2020. those datasets offer a massive repository of API and Permission statistics, for thorough examineDiverse gadget studying fashions had been hired for classification, inclusive of Logistic Regression, “Support Vector Machine, k-Nearest Neighbors (KNN), Random Forest, Decision Tree”, and a Stacking Classifier that merges Random Forest and Multi-Layer Perceptron (MLP) with LightGBM. The mission entails comprehensive dataset preprocessing, model education, and stringent performance assessment. This thorough technique seeks to create very efficient models for the detection of Android malware. This assignment advances cellular security through turning in successful version outcomes, hence enhancing our capability to hit upon and neutralize Android malware attacks efficiently. The venture's effects are sizable for mobile security professionals and researchers. This paintings offers a Stacking Classifier that integrates the blessings of Random forest, "Multi-Layer Perceptron (MLP)", and LightGBM models to enhance function extraction. This ensemble learning method markedly enhances predictive accuracy. furthermore, we created an intuitive Flask framework linked with SQLite for secure registration, login, and consumer checking out. This optimized device helps input submission and prediction acquisition, improving the project's robustness and practicality for powerful person engagement.

“Index terms *- co-existence, FP-growth, machine learning, malware”.*

1. “INTRODUCTION”

The smartphone marketplace is experiencing tremendous boom. The ICD studies [1] estimates that via 2024, global every year mobile cellphone sales will exceed 351 million gadgets. Android is the preeminent cellular operating device, boasting over 2.5 billion

 https://doi.org/10.5281/zenodo.15165584

lively customers in more than 190 countries. The extensive functionalities furnished through smartphones and the growing sort of sports finished through customers, consisting of social networking, on line banking, and gaming, have engendered considerable worries concerning device safety and private privatenessDue to Android's For malware developers, open source nature is relatively easy to build Android malware packages that run attacks and cause considerable damage. The impact of Android malware increases in modern life. Starting in January 2021, Google Play Store, Android Packages Market, consists of over 3.43 million apps. numerous third-birthday celebration and non-Google systems, along with AppBrain and AppChina, are emerging, supplying programs for consumer down load. applications from third-party markets provide a good sized threat of being harmful and are neither monitored nor determined. Allix et al. [6] indicated that 22% of Google Play packages were categorized as harmful, at the same time as 50% of AppChina programs were particular as dangerous.

Security researchers have employed different defensive approaches to discover and stop dangerous applications. Signature-based totally malware detection works as a fundamental detection method that functions with the aid of evaluating patterns among previously diagnosed packages and new programs. it is effective towards regarded malware but can be without difficulty circumvented by way of obfuscation strategies or unusual software. Zhou and Jiang [7] assessed the efficacy of modern signature-based totally anti-malware scanners by checking out four wonderful cell safety applications towards over 1200 Android programs. The findings indicated that current anti-malware applications are unable to perceive disguised or repackaged Trojan packages. Similarly, Scott [8] employed an obstasion strategy Research used ten harmful applications that belonged to different malicious software families. The identifiers failed to detect either of the malicious applications included in the test.

The system employs machine learning logic to separate Android malware from normal packages through an Android malware patternless method. Machine learning technology is called "educational data" and works effectively by creating models that refer to "educational data," effective, and "educational data." which generates predictions or judgments without explicit programming. Malware detection methodologies utilizing Algorithms for machine learning can be divided into two types: static evaluation and dynamic analysis. Static evaluations include evaluations of Android applications. conducted in a specific environment. In contrast, dynamic assessment entails executing the application in a regulated setting to monitor its performance.

Static analysis employs the static indicators derived from manifestos (e.g., applications or permits), source code (e.g., API calls), and intentions [19,21]. A large number of the retrieved features may be disruptive. numerous permissions are solicited by each benign and malicious packages. consequently, various characteristic selection techniques are employed to identify the maximum pertinent homes that efficiently classify malware packages. Invested functional options Statistical techniques implemented by the algorithm determine how well input variables relate to output variables or squares. Each of my functions receives evaluation through the maximum algorithm. This article is based on the correlation of tasks to identify Android malware.

https://doi.org/10.5281/zenodo.15165584

2. LITERATURE SURVEY

In contemporary society, the majority of individuals own at the least one digital computing device with internet connectivity. The net is beginning to exert impact on our each day lives. further to computer systems and cell gadgets, traditionally standalone device and gadgets are an increasing number of linked to the internet, turning into them intelligent. The vital infrastructure of urban regions, healthcare, and other sectors (SCADA) has been integrated with the net to enhance its intelligence. The expansion of the net facilitates a greater handy human life.

Researchers have lately all started to discover the capacity of machine-learning strategies for powerful malware detection across sizeable software datasets. numerous promising consequences were documented inside the literature, with diverse methodologies evaluated thru what we refer to as laboratory validation scenarios. [6]This studies reevaluates the goal of malware detection to take a look at whether or not laboratory validation scenarios yield reliable insights concerning the efficacy of malware detectors in actual-international environments, from time to time referred to as "in the wild." consequently, we have evolved more than one device gaining knowledge of classifiers that make use of a collection of features derived from the packages' "control flow Graphs (CFGs)". We make use of a large dataset along with over 50,000 Android packages gathered from places in which 5bf1289bdb38b4a57d54c435c7e4aa1c methodologies have curated their data. Our approach demonstrates advanced performance compared to currentcutting machine learning-based techniques in laboratory settings. but, this elevated performance does not correspond to advanced performance in actual-international situations. The discovered performance disparity—F-measures declining from over zero.nine in managed environments to underneath 0.1 in actual-world settings—elicits a critical inquiry: How do 5bf1289bdb38b4a57d54c435c7e4aa1c methodologies perform in realistic packages?

The giant use and recognition of smartphones have appreciably expanded the proliferation of cellular malware, mainly on widespread systems like Android [10,15,18]. Given their increasing growth, there's an urgent necessity to formulate suitable answers. nevertheless, our defensive ability is notably hindered by means of the insufficient comprehension of those novel mobile malware and the absence of prompt get admission to to pertinent samples. Seven focus on the Android platform, aiming to categorize existing Android malware. Over 12 months, we recorded over 1,200 malware cases, from samples collected in August 2010 to the final rehearsal in October 2011. Furthermore, we are conducting a prompt investigation into them. From multiple perspectives, their installation tactics, activation mechanisms, and the characteristics of the deleterious payloads they provide. Representative families hold relevance, and subsequent evolutionary studies indicate that they rapidly evolve to evade detection by contemporary cellular antiviral methods. Our analysis of four pertinent mobile security programs indicates that the most effective circumstance detects 79.6% of threats, whilst the suboptimal scenario recognizes only 20.2% of internal data sets. These results necessitate the uneven advancement of the secondary technological anti-mobile-malware technology.

With the annual increase of smartphone users reaching billions, people now keep personal and sensitive facts on their devices, growing a extensive possibility for

https://doi.org/10.5281/zenodo.15165584

hackers to compromise this data. we've offered a technique for detecting Android malware the use of the analysis of permissions and APIs. we've got produced two categories of characteristic vectors, special as not unusual and combined characteristic vectors. We achieved an accuracy of 97.25% for commonplace features and ninety six.56% for composite functions using logistic regression. additionally, to reduce the schooling and trying out period of type, we've got streamlined the function set to 131 by using getting rid of low variance capabilities, ensuing in an accuracy of 95.87%.

Malicious apps threaten Android security. The expanding number and style of these apps makes conventional security useless, leaving Android smartphones vulnerable to new infections. [13] This article presents Drabin, a lightweight Android malware detection approach that identifies hazardous applications on mobile devices simultaneously. on the grounds that confined sources prevent run-time application monitoring, DREBIN plays a radical static analysis to capture as many features as viable. because those tendencies are included in a joint vector area, ordinary malware styles can be robotically discovered and used to give an explanation for our technique's choices. Drebin outperformed other predecessor algorithms, figuring out 94% of malware with a touch fake tremendous price in an evaluation the usage of 123,453 programs and 5,560 malware samples [7,13,38].the explanations highlight relevant malware capabilities. common analysis time on five common smartphones is 10 seconds, making it desirable for checking downloaded apps proper at the device.

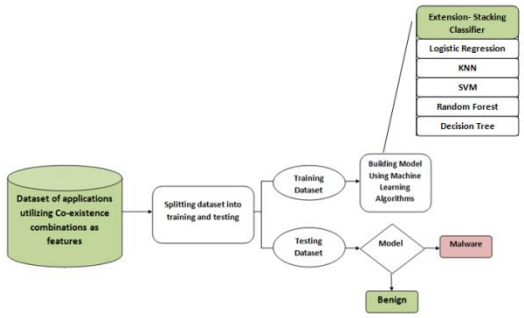
3. METHODOLOGY

i) Proposed Work:

a specific machine learning model for Android malware detection suggests how co-present permissions and APIs distinguish malware from innocuous applications. It outperforms Drebin, CIC_MALDROID2020, and Malgenome models in accuracy [9,13,25]. It pursuits to improve Android security and privateness. The Stacking Classifier in this paper makes use of Random forest, "Multi-Layer Perceptron (MLP)", and LightGBM models to improve characteristic extraction. Ensemble learning greatly improves prediction accuracy. we've created an easy-to-use Flask framework with SQLite for cozy signup, signin, and testing. This simplified technique facilitates enter and prediction retrieval, making the challenge more resilient and person-pleasant.

ii) System Architecture:

The system design begins with a dataset of Android apps with residences generated via co-lifestyles combos[4,5,7]. This dataset is then break up into training and check units. system mastering models (knn, svm, rf, dt, lr, and extension-stacking classifier) start with the schooling set. those models learn to perceive dangerous and benign apps. After that, the check set is used to assess those models' potential to classify fresh packages.



“Fig 1 Proposed architecture”

https://doi.org/10.5281/zenodo.15165584

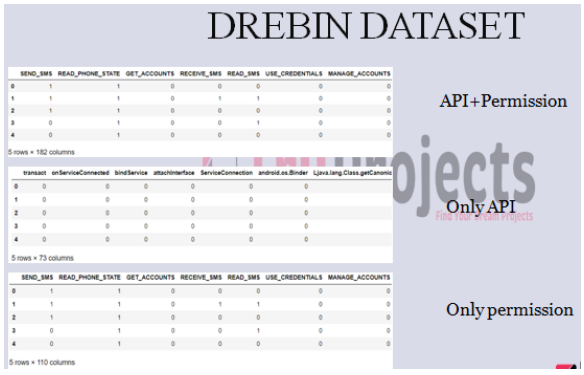
iii) Dataset collection:

DREBIN:

• Drebin is a famous Android malware series. It has several Android apps, both proper and awful. To create powerful fashions for Android malware detection, its scale and variety make it a famous benchmark dataset [9,13,25].

• We utilized the Drebin dataset with these function combinations.

• we are offering the top 5 rows of the data for each characteristic mixture. We can also take a look at the quantity of columns present.

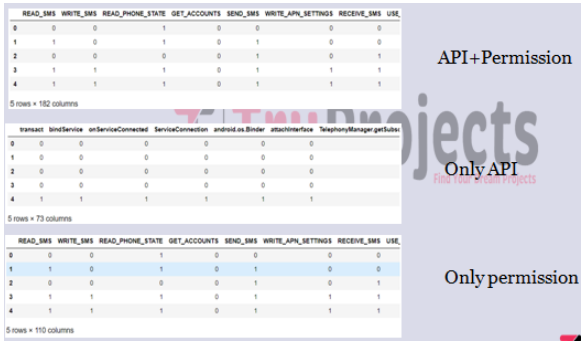


"Fig 2 Drebin dataset"

MALGENOME:

• Malgenomer encompasses Android applications, with a particular emphasis on malware specimens. It comprises several versions of Android malware. This study and model enhance Drabin by providing a curated collection of Android malware samples for model building.

• Malgenom data records were used together with several distinctive combinations [917, 36]. You can view five optimal data records for each functional unit and look at the number of columns in any instance.



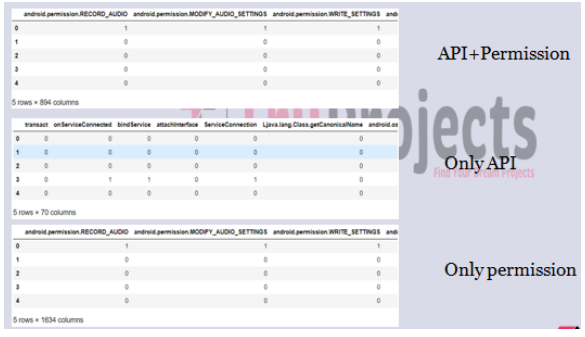
"Fig 3 Malgenome"

CIC_MALDROID2020:

• CIC_MALDROID2020 is obtainable by way of the "Canadian Institute for Cybersecurity (CIC)" and is recognized for its sizeable size, recentness, diversity, and thoroughness.

• Likewise, we utilized the CIC_MALDROID2020 dataset with those characteristic combos.

• we are providing the top 5 rows of the records for every function aggregate here. We may additionally have a look at the quantity of columns gift.



"Fig 4 CIC_MALDROID2020"

iv) Data Processing:

https://doi.org/10.5281/zenodo.15165584

Data processing transforms raw data into a valuable asset for organizations. Statistical researchers typically engage in the manipulation of data, aggregation of datasets, organization, cleansing, validation, assessment, and transformation of information into comprehensible codecs including graphs or reports. Data processing may be executed via 3 methods: manual, mechanical, and digital. The purpose of decoration involves fact cost reduction and supports decision-making processes. The dedicated system enables companies to decorate their operational processes and execute fast strategic decisions. The technological processing of facts through automation helps encompassing laptop software program development, is huge. It may also convert great variations of massive facts, especially huge facts, into huge insights for optimum control and decision-making.

v) Feature selection:

The functional option variant is the most prevalent, non-recessionary, and pertinent facilities that delineate a system for enhancement. It is critical to methodically lessen the dataset`s size, regardless of the breadth and version of the dataset persevering with to expand. The important goal of practical picks is to decorate the efficacy of a prognostic version at the same time as minimizing computational fees related to modeling.

Characteristic preference, a important factor of function engineering, includes figuring out the maximum regularly occurring functions for enter into gadget mastering algorithms. Characteristic need strategies are applied to lessen the variety of enter variables with the aid of using disposing of reproduction or extraneous trends, therefore decreasing the gathering of abilties to the ones maximum applicable to the device mastering version. The key blessings of carrying out function choice in advance, as opposed to permitting the gadget mastering version to independently select the maximum enormous attributes.

vi) Algorithms:

Logistic Regression it is a type method that estimates the chance of an enter being associated with a particular category. The sigmoid function is utilized to transform enter information into a possibility score starting from zero to at least one, with a threshold carried out to categorize the input into one in all two or more training depending on this opportunity. The model acquires coefficients for the duration of training to optimally align with the data and reap particular classifications.

```
from sklearn.linear_model import LogisticRegression
#from sklearn.pipeline import Pipeline

# instantiate the model
log = LogisticRegression()

# fit the model
log.fit(X_train,y_train)
y_pred = log.predict(X_test)
```

“Fig 5 Logistic regression”

A **Support Vector Classifier** (SVC) A device is a mastering version that determines the gold standard hyperplane to segregate numerous statistics instances, at the same time as maximizing the margin among them. It determines the essential help vector to allow unique classifications and robustly substances it for each binary and multi-magnificence type applications.

https://doi.org/10.5281/zenodo.15165584

```
# Support Vector Classifier model
from sklearn.svm import SVC
svc = SVC()

# fit the model
svc.fit(X_train,y_train)
y_pred = svc.predict(X_test)
```

“Fig 6 SVM”

K-Nearest Neighbors (KNN) it is a multifaceted machine learning technique hired for type and regression programs. It identifies the ok nearest information points to a designated input and generates predictions with the aid of majority vote or common. KNN is non-parametric and easy to construct; but, it could be sensitive to the choice of okay and might underperform in high-dimensional facts with out preprocessing [30].

```
from sklearn.neighbors import KNeighborsClassifier
neigh = KNeighborsClassifier(n_neighbors=3)

# fit the model
neigh.fit(X_train,y_train)
y_pred = neigh.predict(X_test)
```

“Fig 7 KNN”

The ensemble learning method Random Forest unites many preference models which generate predictions by averaging their outcomes. This technique builds a collection of selected trees for randomly picked subsets of data before achieving predictions based on mean calculations.Random Forest is an ensemble method that enhances accuracy while offering adaptable solutions and robust performance for classification and regression tasks.

```
# Random Forest Classifier Model
from sklearn.ensemble import RandomForestClassifier

# instantiate the model
forest = RandomForestClassifier(n_estimators=10)

# fit the model
forest.fit(X_train,y_train)
y_pred = forest.predict(X_test)
```

“Fig 8 Random forest”

A **Decision Tree** is a mechanical learning version that determines the results by recursive distribution data of subquants according to their best features, for classification or prediction purposes. Each knot means a feature, and each branch defines a hierarchical framework that makes it easy to understand and green in various sports.

```
# Decision Tree Classifier model
from sklearn.tree import DecisionTreeClassifier

# instantiate the model
tree = DecisionTreeClassifier(max_depth=30)

# fit the model
tree.fit(X_train, y_train)

y_pred = tree.predict(X_test)
```

“Fig 9 Decision tree”

A Stacking Classifier is an ensemble learning technique that integrates the predictions of numerous base models, along with "Random forest (RF)", "Multi-Layer Perceptron (MLP)", and LightGBM, to produce a greater particular very last prediction. It utilizes the advantages of those separate fashions to decorate average predictive efficacy. initially, the base fashions are educated at the education statistics, and their predictions in the end function enter capabilities for a meta-learner, which learns to integrate these

https://doi.org/10.5281/zenodo.15165584

predictions to offer the very last output. Stacking is an powerful technique applied to enhance predictive accuracy and is regularly implemented in machine learning across diverse packages.

```
estimators = [('rf', RandomForestClassifier(n_estimators=10)),('mlp', MLPClassifier(random_state=1, max_iter=300))]

clf1 = StackingClassifier(estimators=estimators, final_estimator=LGBMClassifier())
# fit the model
clf1.fit(X_train, y_train)
# fit the model
y_pred = clf1.predict(X_test)
```

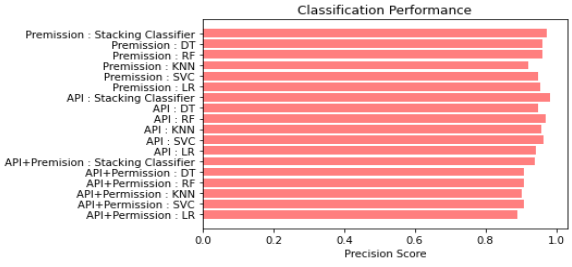
"Fig 10 Stacking classifier"

4. EXPERIMENTAL RESULTS

Precision: The accuracy establishes the ratio between correctly identified instances among those instances which received an effective designation. The mathematical formula to calculate precision appears as follows:

"Precision = True positives/ (True positives + False positives) = TP/(TP + FP)"

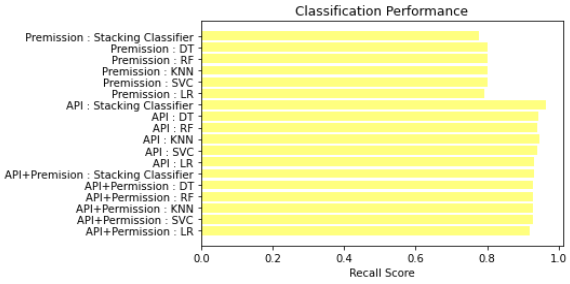
$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$



"Fig 11 Precision comparison graph"

Recall: The Idea Machine is a computational framework in learning that evaluates a model's capacity to encompass all pertinent instances of a selected excellence. This segment is anticipated to yield effective observations for comprehensive authenticity, and identifying occurrences inside a chosen category will provide insights into the efficacy of a variant.

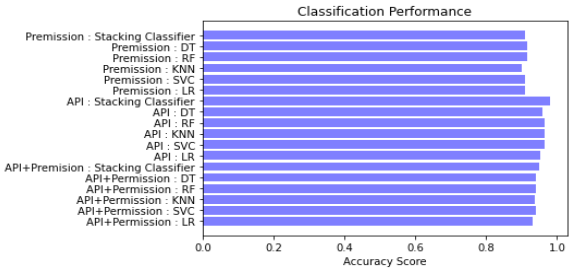
$$Recall = \frac{TP}{TP + FN}$$



"Fig 12 Recall comparison graph"

Accuracy: Accuracy is the percentage of actual forecasts for a particular company and evaluates the overall integrity of the model composition.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

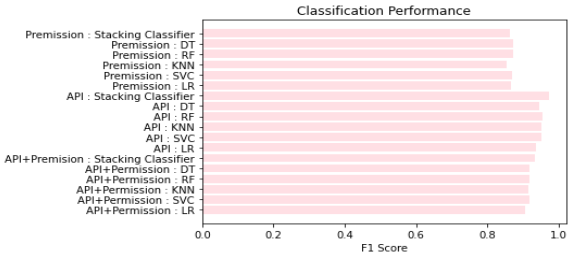


"Fig 13 Accuracy graph"

F1 Score: F1 Point is a metric that combines accuracy and recall, taking into account both false positives and false negative statements and provides a balanced rating suitable for imbalanced data records.

https://doi.org/10.5281/zenodo.15165584

$$\text{F1 Score} = 2 * \frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} * 100$$



"Fig 14 F1Score"

	ML Model	Accuracy	Precision	f1_score	Recall
0	API+Permission : LR	0.99	0.98	0.98	0.99
1	API+Permission : SVC	0.99	0.98	0.99	1.00
2	API+Permission : KNN	0.97	0.94	0.97	0.99
3	API+Permission : RF	0.97	0.97	0.97	0.97
4	API+Permission : DT	0.95	0.97	0.95	0.94
5	**API+Permission : Stacking Classifier**	**1.00**	**1.00**	**1.00**	**1.00**
6	API : LR	0.95	0.94	0.95	0.97
7	API : SVC	0.96	0.94	0.96	0.97
8	API : KNN	0.94	0.93	0.94	0.95
9	API : RF	0.94	0.94	0.94	0.94
10	API : DT	0.95	0.95	0.94	0.94
11	**API : Stacking Classifier**	**0.97**	**0.98**	**0.97**	**0.96**
12	Premission : LR	0.95	0.93	0.95	0.97
13	Premission : SVC	0.95	0.92	0.95	0.97
14	Premission : KNN	0.94	0.90	0.94	0.99
15	Premission : RF	0.95	0.93	0.95	0.97
16	Premission : DT	0.95	0.94	0.95	0.95
17	**Premission : Stacking Classifier**	**0.99**	**0.98**	**0.99**	**1.00**

"Fig 15 Performance Evaluation"

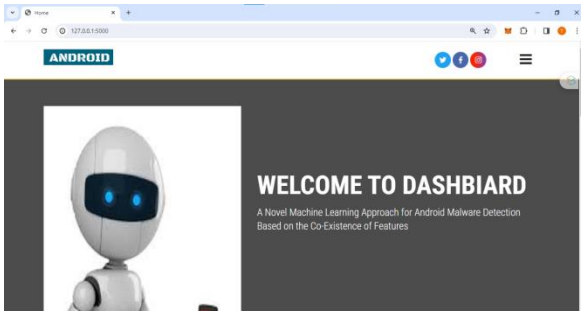
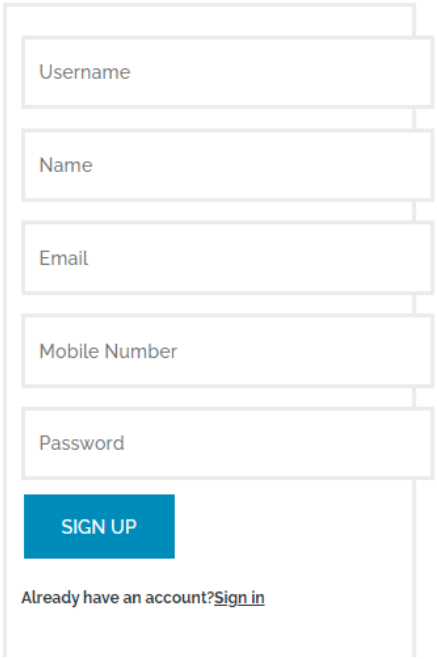
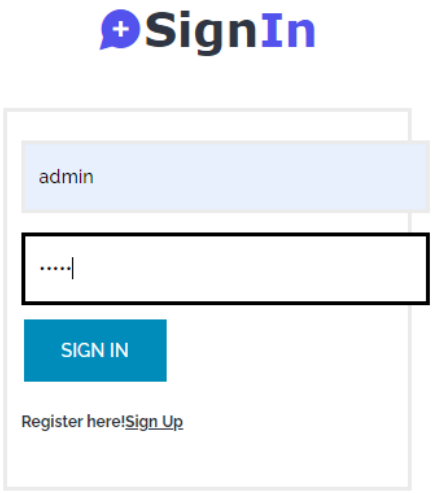


Fig 16 Home page

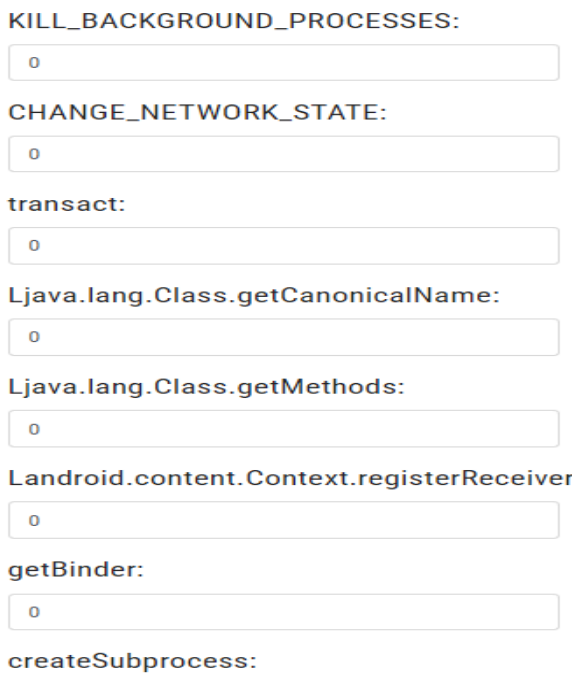


"Fig 17 Signin page"

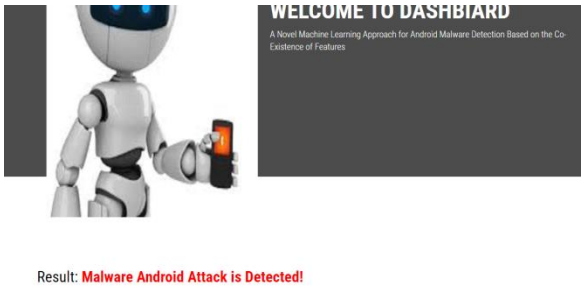


"Fig 18 Login page"

https://doi.org/10.5281/zenodo.15165584



"Fig 19 User input"



"Fig 20 Predict result for given input"

5. CONCLUSION

The test illustrated the efficacy of machine learning models in figuring out Android malware. those models established strong proficiency in detecting fraudulent applications, which is important for safeguarding Android users. The stacking classifier proven superior efficacy compared to separate algorithms. the mixing of different fashions improved normal detection accuracy, demonstrating the efficacy of ensemble studying. Flask and SQLite facilitate an intuitive interface, improving accessibility. Design user testing, specify verification, and continuous models facilitate predictions, yielding practical use and widespread appeal. The assignment emphasised the importance of integrating each API and Permission functionalities. This combination become deemed crucial for enhancing malware detection, highlighting the significance of using numerous static capabilities inside the evaluation. The efficacy of machine learning models differed amongst numerous datasets, together with Drebin, Malgenome, and CIC_MALDROID2020 [36]. This highlights the significance of meticulous dataset selection and comprehension for growing specific detection models. The fashions attained an equilibrium among precision and the discount of false positives. this is crucial as it guarantees that during malware detection, actual applications are not erroneously diagnosed as threats, subsequently minimizing user infection. The effects of this assignment have substantial ramifications. safety professionals can hire those advanced detecting techniques to augment cybersecurity. App builders can beautify the security in their applications towards capacity risks, whilst give up customers experience progressed safety towards Android malware, ensuing in a more secure cellular experience.

6. FUTURE SCOPE

subsequent study might also focus on improving the real-time detection capabilities of the suggested system by using the non-stop monitoring and evaluation of dynamic variables. this would allow the device to react more correctly to emerging Android malware threats. Investigating methods to pick out the maximum pertinent dynamic elements for malware detection can result in greater efficient and specific fashions. strategies for feature selection, which

https://doi.org/10.5281/zenodo.15165584

includes mutual statistics and recursive feature removal, want investigation. enhancing the machine to perceive behavioral anomalies in Android applications would possibly provide an extra layer of protection. This involves recognizing anomalies from anticipated conduct, which may additionally suggest the presence of malware. 7 As novel forms of Android malware rise up, the system may be engineered to adapt and refresh its models and detection methodologies. consistently integrating new threat intelligence and refreshing the system is essential for sustained efficacy. enhancing the machine's abilties to consist of go-platform malware detection, inclusive of iOS and different cellular running systems, can provide a greater comprehensive answer for cellular protection.

REFERENCES

[1] H. Menear. (2021). IDC Predicts Used Smartphone Market Will Grow 11.2% by 2024. Accessed: Oct. 30, 2022. [Online]. Available: https://mobile-magazine.com/mobile-operators/idc-predicts-usedsmartphone-market-will-grow-112-2024?page=1

[2] D. Curry. (2022). Android Statistics. Accessed: Oct. 30, 2022. [Online]. Available: https://www.businessofapps.com/data/android-statistics/

[3] O. Abendan. (2011). Fake Apps Affect Android Os Users. Accessed: Oct. 30, 2022. [Online]. Available: https://www.trendmicro.com/ vinfo/us/threat-encyclopedia/web-attack/72/fake-apps-affect-android-osusers

[4] C. D. Vijayanand and K. S. Arunlal, ‘‘Impact of malware in modern society,’’ J. Sci. Res. Develop., vol. 2, pp. 593–600, Jun. 2019.

[5] M. Iqbal. (2022). App Download Data. Accessed: Oct. 30, 2022. [Online]. Available: https://www.businessofapps.com/data/app-statistics/

[6] K. Allix, T. Bissyand, Q. Jarome, J. Klein, R. State, and Y. L. Traon, ‘‘Empirical assessment of machine learning-based malware detectors for android,’’ Empirical Softw. Eng., vol. 21, pp. 183–211, Jun. 2016.

[7] Y. Zhou and X. Jiang, ‘‘Dissecting Android malware: Characterization and evolution,’’ in Proc. IEEE Symp. Secur. Privacy, May 2012, pp. 95–109.

[8] J. Scott. (2017). Signature Based Malware Detection is Dead. Accessed: Oct. 30, 2022. [Online]. Available: https://icitech.org/wpcontent/uploads/2017/02/ICIT-Analysis-Signature-Based-MalwareDetection-is-Dead.pdf

[9] Q. M. Y. E. Odat. Accessed: Dec. 27, 2022. [Online]. Available: https://github.com/esraa-cell28/a-novel-machine-learning-approach-forandroid-malware-detection-based-on-the-co-existence

[10] S. R. Tiwari and R. U. Shukla, ‘‘An Android malware detection technique based on optimized permissions and API,’’ in Proc. Int. Conf. Inventive Res. Comput. Appl. (ICIRCA), Jul. 2018, pp. 258–263.

[11] (2018). Dex2jar—Tools To Work With Android.dex & Java.Class Files. Accessed: Oct. 30,

https://doi.org/10.5281/zenodo.15165584

2022. [Online]. Available: https://kalilinuxtutorials.com/dex2jar-android-java/

[12] Androzoo. Accessed: Jul. 30, 2022. [Online]. Available: https://androzoo.uni.lu/

[13] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, and K. Rieck, ‘‘Drebin: Effective and explainable detection of Android malware in your pocket,’’ in Proc. NDSS, Feb. 2014, pp. 23–26.

[14] Virusshare. accessed: Jul. 30, 2022. [Online]. Available: https://virusshare.com/

[15] H. Cheng, X. Yan, J. Han, and C.-W. Hsu, ‘‘Discriminative frequent pattern analysis for effective classification,’’ in Proc. IEEE 23rd Int. Conf. Data Eng., Apr. 2007, pp. 716–725.

[16] M. Parkour. Contagio Mini-Dump. accessed: Jul. 30, 2022. [Online]. Available: http://contagiominidump.blogspot.it/

[17] Malgenome Project. accessed: Jul. 30, 2022. [Online]. Available: http://www.Malgenomeproject.org

[18] C.-F. Tsai, Y.-C. Lin, and C.-P. Chen, ‘‘A new fast algorithms for mining association rules in large databases,’’ in Proc. IEEE Int. Conf. Syst., Man Cybern. San Francisco, CA, USA: Morgan Kaufmann, Oct. 1994, pp. 487–499.

[19] A. Lab. (2017). Amd Dataset. Accessed: Oct. 30, 2022. [Online]. Available: https://www.kaggle.com/datasets/blackarcher/malware-dataset

[20] V. Avdiienko, ‘‘Mining apps for abnormal usage of sensitive data,’’ in Proc. IEEE/ACM 37th IEEE Int. Conf. Softw. Eng., vol. 1, May 2015, pp. 426–436.

[21] Y. Aafer, W. Du, and H. Yin, ‘‘Droidapiminer: Mining api-level features for robust malware detection in android,’’ in Security and Privacy in Communication Networks, T. Zia, A. Zomaya, V. Varadharajan, and M. Mao, Eds. Cham, Switzerland: Springer, 2013, pp. 86–103.

[22] V. M. Afonso, M. F. de Amorim, A. R. A. Grégio, G. B. Junquera, and P. L. De Geus, ‘‘Identifying Android malware using dynamically obtained features,’’ J. Comput. Virology Hacking Techn., vol. 11, no. 1, pp. 9–17, 2015.

https://doi.org/10.5281/zenodo.15165584