Detection of Android Malware Using Multiple Linear Regression Models-Based Classifiers

M. Parkavi

Assistant professor ,Department of Computer science
Christ Arts and Science College, Kilachery
parkaviberni26@gmail.com

ABSTRACT

Android is an operating system which presently has over one billion active druggies for all their mobile bias, with a request impact that's impacting an increase in the quantum of information that can be attained from different druggies, data that have motivated the development of malware by cyber culprits. In this study, a frame for Android malware discovery grounded on warrants is presented. This frame uses multiple direct regression styles. operation warrants, which are one of the most critical structure blocks in the security of the Android operating system, are uprooted through static analysis, and security analyzes of operations are carried out with machine literacy ways. Grounded on the multiple direct regression ways, two classifiers are proposed for authorization- grounded Android malware discovery. These classifiers are compared on four different datasets with introductory machine literacy ways similar as support vector machine, k- nearest neighbor, Naive Bayes, and decision trees. In addition, using the bagging system, which is one of the ensemble literacy, different classifiers are created, and the bracket performance is increased. As a result, remarkable performances are attained with bracket algorithms grounded on direct regression models without the need for veritably complex bracket algorithms.

KEYWORDS: *Ensemble literacy, direct regression, machine literacy, malware analysis, authorization grounded android malware discovery, static analysis.*

INTRODUCTION:

Preface When the first mobile phones were considered, generally speaking or short communication deals were carried out with mobile phones in diurnal life. still, with mobile phones used moment, remarkable deals similar as banking deals, social media use, and particular data storehouse take place. Because of these essential processes, mobile bias are the main target of malware inventors. Android is an open- source Linux- grounded mobile operating system. Since it's open- source and free, mobile device manufacturers prefer this operating system on their bias. thus, the maturity of the request consists of Android bias. According to Statista's data, 30 of the request in the fourth quarter of 2010 comported of the Android operating system.

In the alternate quarter of 2018, 88 of the request was Android operating systems. In addition to Android being an open- source operating system, it's veritably flexible for druggies that operations are handed to bias similar as other stores or third- party operations piecemeal from the sanctioned operation stores. For this reason, Android is constantly preferred by numerous people around the world. Although operations from unofficial operation depositories or third- party operation inventors are veritably profitable for druggies, it shouldn't be ignored that some of these operations are malware. Apps in sanctioned app depositories are precisely anatomized and published in app depositories.

Still, malware is common indeed in sanctioned operation depositories. In the exploration conducted by Wangetal., further than 6 million operations downloaded from 17 operation stores are estimated. While 16 of these stores are extensively used in China, the first place is Google Play. In general, it's revealed that Google Play is more dependable than other operation stores. still, it's possible to see malware in nearly all stores. While 1 million new malware were detected in the first six months of 2015,1.85 million new malware were detected in the first six months of 2019. Despite all the preventives, there's a remarkable increase in the number of vicious software. For this reason, both experimenters and companies working on computer security offer new approaches for detecting mobile malware.

In this study, a machine literacy- grounded Android malware discovery system is developed, in which operation warrants, which have an important place in Android security, are used as attributes. After an operation is installed on the device, numerous warrants are requested from the stoner. While the operation is running in the background, the operation can show its vicious point in line with the warrants given by the stoner. thus, druggies should pay attention to the requested warrants. In this study, the warrants requested by the operations are estimated with machine literacy models, and it's decided whether the operation is malware or not.

A)RELATED WORKS:

In recent times, numerous studies have been conducted to descry Android malware using machine literacy or deep literacy approaches. Discovery styles differ according to the way in which the features used in machine literacy or deep literacy approaches are attained. These are generally stationary, dynamic, and mongrel analysis ways. In dynamic analysis, features for machine literacy approaches are attained by running operations on a real or virtual device. In stationary analysis, features are uprooted for machine literacy approaches without running operations. Since operations are run in dynamic analysis, it's challenging to produce the necessary structure. still, they're successful against zero- day attacks.

In stationary analysis, the process is relatively fast since operations aren't run. In addition to static and dynamic analysis ways, there's also a mongrel analysis approach. In this approach, features attained from static and dynamic styles are used together. Some Android malware discovery systems using static, dynamic, and mongrel analysis approaches are as follows In, it was classified 2000 vicious operations conforming of 18 families according to their families. operations were reused through the ditz Sandbox, rooting the most distinctive behavioral features that distinguish vicious families from each other. The attained features were given to a system called online machine literacy, and bracket of malware according to their families is carried out.

In the trials, all of the operations in 7 classes were classified rightly. The class with the smallest performance rate was determined as theandroid.trojan.smskey family.

In, a malware discovery system grounded on dynamic analysis was proposed. In total, further than 12000 operations were estimated. While 4289 of these operations were vicious, 8371 of them were benign. vicious operations were attained from the Drebin dataset, while benign operations were downloaded from Google Play. System calls were uprooted stoutly and used as attributes for machine literacy algorithms. The generation of system calls was handled by the sandbox. What operations do on the operating system was recorded in log lines. therefore, the actions of each operation were formed chronologically. While penetrating system calls, malware wasn't allowed to affect these calls.

In this way, the situation of changing the geste of vicious software was also excluded. Thanks to this point, the proposed system was resistant to simple obfuscation ways, which are frequently seen in malware. point vectors were created by recycling the attained log lines. In the last step, these point vectors were estimated with machine literacy approaches, and bracket of benign and vicious software was carried out. In the bracket phase, machine literacy ways similar as support vector machines(SVM), arbitrary timber(RF), LASSO, and crest regularization were used. The stylish performance was attained from the RF algorithm.

The authors offered two different approaches grounded on stationary analysis by making use of machine literacy approaches. In the first approach, operation warrants were uprooted with static analysis. In the alternate approach, source law analysis was done with the bag- of- words model. It was stated that the computational cost of the first

approach is fairly low compared to the alternate approach. A large number of trials were carried out using both clustering and bracket algorithms. C4.5 decision tree, RF, Bayes networks, successional minimum optimization(SMO), repeated incremental pruning(JRip), logistic regression were some of the algorithms used.

In addition, models grounded on bagging ways were developed by combining bracket algorithms. Machine literacy algorithms were run on the M0Droid dataset, which consists of 200 vicious and 200 benign Android operations. The loftiest performance attained in the authorization- grounded approach was attained with the SMO algorithm. This performance was0.879 grounded on the f- measure metric. By trying different bagging ways, this success was increased up to0.894. The source law analysis, the loftiest performance was achieved with the SMO algorithm. This performance was0.951 according to the f- measure metric. By trying different bagging ways, this success was increased up to0.9560.

In, the authors handed the discovery of Android malware with a dynamic analysis fashion. In the dynamic analysis phase, the geste of the operations was anatomized by considering the system calls. The proposed armature was called ANDROIDETECT. ANDROIDETECT was a machine literacy- grounded Android malware discovery system that enables instant attack discovery. The bracket result of the proposed discovery system has a low false-positive rate, thanks to the creation of effective point vectors. point vectors were created by rooting the system call function. Bracket algorithms also estimated these point vectors. The study used two different bracket algorithms, naive Bayes(NB) and J48 decision trees. trials were carried out with 100 benign and 100 vicious operations. The result from the NB classifier is0.825 according to the f- measure metric. In discrepancy, the result attained from the J48 classifier is0.86 according to the f- measure metric. In, 1233 Android malware were classified according to types. In total, 28 different types of Android malware were classified according to their types. operation warrants are given as input to machine literacy algorithms. Some warrants were under the veritably dangerous group, while some warrants were under the fairly less dangerous group. To digitize these differences and ameliorate the performance of bracket algorithms, the authors proposed a fashion they call an Extremely Randomized Tree.

The proposed system also satisfied the point selection task. Six different bracket algorithms were used in the study. These are SVM, ID3 decision trees, RF, neural networks, nearest neighbor, and bagging algorithms. The stylish bracket result is attained with the RF algorithm. The bracket affect attained with the RF is95.97. In, a authorization- grounded Android malware discovery system grounded on machine literacy algorithms was presented. With the system called significant authorization identification(SIGPID), rather of using all warrants, it was handed to choose the warrants that will grease the separation of vicious software from vicious software. With the proposed system, 135 warrants were reduced to 22 warrants. When bracket was made with 22 warrants, further successful and briskly results are attained.

In addition, it was emphasized that over 90 bracket success was achieved with the SVM in the study. In 31185 benign and 15336 vicious Android operations were used. warrants and API calls were uprooted as attributes in the malware discovery system called MalPat. RF algorithm was used in the bracket phase of the study. When the experimental results were examined, a bracket success rate of98.24 was attained according to the f- measure. An Android malware discovery system grounded on deep neural networks(DNN) was proposed. operation warrants uprooted using the static analysis fashion were used as attributes. In the study, expansive trials compared deep neural networks with numerous traditional machine learning approaches.

II. METHODOLOGY:

A.DATA PREPROCESSING AND PREPARATION:

Android Package Kit(APK) is known as the package train format used by the Android operating system to distribute and install mobile operations. thus, APK lines are demanded in the Android operating system. APK lines can be allowed of as compressed lines. In general, these lines include operation source canons, operation warrants, image and videotape lines in operations.

Android operations are generally written using the Java programming language. also, Java source canons are collected and converted into byte canons. Considering computers with a Windows or Linux- grounded operating system on which the Java virtual machine is installed, these collected byte canons are converted into a structure that can be run on the applicable operating system. still, byte canons can not be run directly in the Android operating system. thus, bytecodes are converted to executable Dalvik bytecodes by performing one further operation on bytecodes. therefore, these Dalvik bytecodes can now be run with the help of the Dalvik Virtual Machine. As a result, the written operations are run on the device. rooting information from APK lines is the reverse of compendium. This process is called Decompilation.

PROPOSED CLASSIFIERS :

We originally give classifiers attained from direct regression in Section II- B1. also, we show combining the stylish algorithms according to the bagging fashion in Section II- B2.

1)LINEAR Regression- Grounded CLASSIFIERS

The linear regression fashion is a constantly used system in working estimation problems. It's grounded on the proposition that samples in the same class belong to the same direct subspace and can be represented by a direct equation. Equation 1 shows the simple linear regression model.

$$y = \beta 0 + \beta 1X + \varepsilon \qquad (1)$$

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="com.android.alarm.permission.SET_ALARM" />
<uses-permission android:name="android.permission.VIBRATE" />
```

Some permissions appearing in AndroidManifest.xml.

In Equation 1, y is called the dependent variable, and X is called the independent variable. The point where the line intersects the y-axis is β0, while β1 represents the regression coefficient. Finally, ε represents the error of the obtained estimate. Equation 1 is known as simple linear regression since it contains only the independent variable X. If there is more than one independent variable affecting the Equation 1, it is called multiple linear regression. The multiple regression model is given in Equation 2. Considering the Equality 2, there are many independent variables consisting of X1, X2…,Xn.

Considering the problem addressed in this study, while attributes, in other words, permissions, represent the independent variable, y represents the class of an application. A multiple linear regression model is needed because a large number of application permissions are used as attributes. In Table 1, the type of application is shown as benign or malicious. Since the systems of equations are solved in linear regression, operations are performed by using 1 instead of benign and 0 instead of malicious.

Suppose a dataset consists of N applications and M permissions (p1, p2, . . , pM) obtained from these applications. A system of equations can be created when there is a linear relationship between permissions and applications, as shown in Equation 3.

$$y = \beta 0 + \beta 1X1 + \beta 2X2 + \ldots + \beta nXn + \varepsilon \qquad (2)$$

$$\begin{cases} y'_1 = \beta_0 + \beta_1 p_{1,1} + \beta_2 p_{1,2} + \ldots + \beta_M p_{1,M} \\ y'_2 = \beta_0 + \beta_1 p_{2,1} + \beta_2 p_{2,2} + \ldots + \beta_M p_{2,M} \\ y'_3 = \beta_0 + \beta_1 p_{3,1} + \beta_2 p_{3,2} + \ldots + \beta_M p_{3,M} \\ \vdots \\ y'_N = \beta_0 + \beta_1 p_{N,1} + \beta_2 p_{N,2} + \ldots + \beta_M p_{N,M} \end{cases} \quad (3)$$

In Equation 3, y'1 , y '2 , . . . , y 'N represents the result of linear combinations of permissions (p1, p2, . . . , pM).βi shows the effect of permissions on y'1 , y'2 , . . . , y'N values.

In Equation 3, it is aimed to find the appropriate βi($1 \leq i \leq M$) parameter for linear regression model. The actual class values (y1, y2, . . . , yN) will be approximately equal to y'1 , y'2 , . . . , y'N values.

The mean square error is generally used to measure the quality of the direct regression model. The lower the mean square error, the near the direct regression model will produce to the factual value. thus, in order to gain a good quality regression model, it's necessary to make the mean square error of the model as small as possible. Hence, quality regression models are created by chancing the most applicable βi parameter. Equation 4 shows how the sum of places of errors(SSE) is calculated.

$$\begin{aligned} SSE &= \sum_{j=1}^{N} (y_j - y'_j)^2 \\ &= \sum_{j=1}^{N} (y_j - \beta_0 - \sum_{k=1}^{M} \beta_k p_{j,k})^2 \end{aligned} \quad (4)$$

In order to minimize the SSE function attained in Equation 4, the partial derivations of this function with respect to each of its βi($1 \leq i \leq M$) unknowns must be taken. Since it's aimed to minimize the error, the result of partial derivations is equal to 0. Equation 5 shows partial derivations.

$$\begin{aligned} \frac{\partial SSE}{\partial \beta_0} &= \frac{\partial \sum_{j=1}^{N} (y_j - \beta_0 - \sum_{k=1}^{M} \beta_k p_{j,k})^2}{\partial \beta_0} = 0 \\ \frac{\partial SSE}{\partial \beta_1} &= \frac{\partial \sum_{j=1}^{N} (y_j - \beta_0 - \sum_{k=1}^{M} \beta_k p_{j,k})^2}{\partial \beta_1} = 0 \\ &\vdots \\ \frac{\partial SSE}{\partial \beta_M} &= \frac{\partial \sum_{j=1}^{N} (y_j - \beta_0 - \sum_{k=1}^{M} \beta_k p_{j,k})^2}{\partial \beta_M} = 0 \end{aligned} \quad (5)$$

Equation 6 is attained when partial derivatives are applied according to each of the βi unknowns in Equation 5. A matrix and Y vector shown in Equation 6 can be attained directly from the dataset. Since A and Y are known, the vector β can be set up with A −1 Y operation. Each element of the performing β vector corresponds to βi unknowns, singly.

Eq.(6), as shown at the bottom of the coming runner. As a result of the calculation of the regression portions(βi) in Equation 2, a direct regression model will be attained. When the point vectors attained from the operations are given to this model, as shown in Table 1, the class value of the operation belonging to the point vector is determined. As a result of this calculation, the class value of the applicable operation emerges, not the class marker.

$$\begin{aligned}
&\beta_0 N + \beta_1 \sum_{i=1}^{N} p_{i,1} + \beta_2 \sum_{i=1}^{N} p_{i,2} + \ldots + \beta_M \sum_{i=1}^{N} p_{i,M} = \sum_{i=1}^{N} y_i \\
&\beta_0 \sum_{i=1}^{N} p_{i,1} + \beta_1 \sum_{i=1}^{N} p_{i,1}^2 + \beta_2 \sum_{i=1}^{N} p_{i,1}p_{i,2} + \ldots + \beta_M \sum_{i=1}^{N} p_{i,1}p_{i,M} = \sum_{i=1}^{N} p_{i,1}y_i \\
&\beta_0 \sum_{i=1}^{N} p_{i,2} + \beta_1 \sum_{i=1}^{N} p_{i,1}p_{i,2} + \beta_2 \sum_{i=1}^{N} p_{i,2}^2 + \ldots + \beta_M \sum_{i=1}^{N} p_{i,2}p_{i,M} = \sum_{i=1}^{N} p_{i,2}y_i \\
&\vdots \\
&\underbrace{\beta_0 \sum_{i=1}^{N} p_{i,M} + \beta_1 \sum_{i=1}^{N} p_{i,1}p_{i,M} + \beta_2 \sum_{i=1}^{N} p_{i,2}p_{i,M} + \ldots + \beta_M \sum_{i=1}^{N} p_{i,M}^2 = \sum_{i=1}^{N} p_{i,M}y_i}
\end{aligned}$$

$$\Downarrow$$

$$\underbrace{\begin{bmatrix}
N & \sum_{i=1}^{N} p_{i,1} & \sum_{i=1}^{N} p_{i,2} & \cdots & \sum_{i=1}^{N} p_{i,M} \\
\sum_{i=1}^{N} p_{i,1} & \sum_{i=1}^{N} p_{i,1}^2 & \sum_{i=1}^{N} p_{i,1}p_{i,2} & \cdots & \sum_{i=1}^{N} p_{i,1}p_{i,M} \\
\sum_{i=1}^{N} p_{i,2} & \sum_{i=1}^{N} p_{i,1}p_{i,2} & \sum_{i=1}^{N} p_{i,2}^2 & \cdots & \sum_{i=1}^{N} p_{i,2}p_{i,M} \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
\sum_{i=1}^{N} p_{i,M} & \sum_{i=1}^{N} p_{i,1}p_{i,M} & \sum_{i=1}^{N} p_{i,2}p_{i,M} & \cdots & \sum_{i=1}^{N} p_{i,M}^2
\end{bmatrix}}_{A}
\underbrace{\begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_M \end{bmatrix}}_{\beta}
=
\underbrace{\begin{bmatrix} \sum_{i=1}^{N} y_i \\ \sum_{i=1}^{N} p_{i,1}y_i \\ \sum_{i=1}^{N} p_{i,2}y_i \\ \vdots \\ \sum_{i=1}^{N} p_{i,M}y_i \end{bmatrix}}_{Y} \qquad (6)$$

Algorithm 1 Determining Class Labels With LinRegDroid1

Input: TestData[ ][ ] and β[ ] represent the dataset and the regression coefficients, respectively.

Output: ClassificationLabel[ ] represents the predicted class labels of each tested application.

1.function Classify(TestData()(), β())

2.N1 ←number_of, operations

3. N2 ←number_of, warrants

4. results() ← ∅

5. for i ← 1 to N1 do

6. sum ← 0

7. for j ← 1 to N2 do

8. sum ← sum TestData(i)(j) * β(j)

9. end for

10. results(i) ← β(0) sum

11. end for

12. ClassificationLabel() ← ∅

13 for i ← 1 to N1 do

14.if results(i)> = 0.5 also

15.ClassificationLabel(i) ← 1

16. else

17. ClassificationLabel(i) ← 0

18. end if

19 .end for

20. return ClassificationLabel()

21. end function

Algorithm 2 Determining Class Labels With LinRegDroid2

Input: TestData[ ][ ] and β[ ] represent the dataset and the regression coefficients, respectively.

Output: ClassificationLabel[ ] represents the predicted class labels of each tested application.

1 function Classify(TestData()(), β())

2 N1 ←number_of, operations

3 N2 ←number_of, warrants

4 results() ← ∅

5 for i ← 1 to N1 do

6 sum ← 0

7 for j ← 1 to N2 do

8 sum ← sum TestData(i)(j) * β(j)

9 end for

10 results(i) ← β(0) sum

11 end for

12 ClassificationLabel() ← ∅

13 for i ← 1 to N1 do

14 if abs(0 − results(i))< abs(1 − results(i)) also

15 ClassificationLabel(i) ← 0

16 else

17 ClassificationLabel(i) ← 1

18 end if

19 end for

20 return ClassificationLabel()

21 end function

EXPERIMENTAL SETTINGS:

This section consists of three subsections. In Section III- A, the datasets used are mentioned. In Section III- B, we give further details about compared classifiers with which the proposed bracket approaches. In Section III- C, we present the criteria used to measure the performance of bracket algorithms.

DATASETS USED:

In this study, four different datasets are used. The first dataset is participated by Ali Dehghantanha, one of the authors of study M0Droid. In this dataset, there are 200 benign and 200 vicious operations. When the data preprocessing step in Section II- A is applied to this dataset, 76 native warrants are uprooted as attributes.

The alternate dataset is AMD. There are 1000 vicious and 1000 benign operations in this dataset. The vicious operations in this dataset are attained. Benign operations are downloaded from the APK Pure app store. We prize 102 native warrants from the AMD dataset. The third dataset is participated.

There are 558 operations in total in this dataset. Half of these operations are benign, while the remaining half are vicious. There are 330 attributes in this dataset, conforming of native and custom-made warrants. Eventually, the fourth dataset is participated. There are 7622 operations in total in this dataset. While 6661 of these operations are vicious, 961 of them are benign. This dataset contains 349 attributes conforming of native and custom-made warrants.

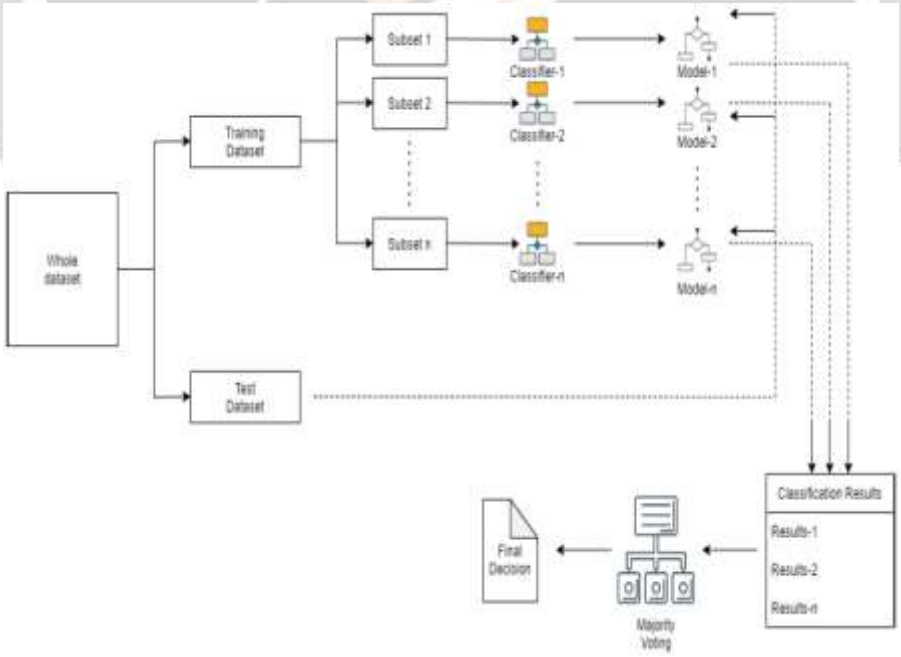

TABLE 6. Results from the M0Droid dataset.

	accuracy(%)	f_measure
LinRegDroid1	82.942	0.8287
LinRegDroid2	82.942	0.8287
KNN	82.69	0.8258
mvmn-NB	77.923	0.7765
mn-NB	77.429	0.7733
rbf-SVM	86.962	0.8673
linear-SVM	86.212	0.8619
DT	86.212	0.8619
Bagging-DT	87.205	0.8712
Ensemble-1	83.74	0.8348
Ensemble-2	89.22	0.8915

TABLE 7. Results from the Arslan' dataset.

	accuracy(%)	f_measure
LinRegDroid1	96.69	0.9172
LinRegDroid2	96.69	0.9172
KNN	96.54	0.9126
mvmn-NB	93.96	0.8571
mn-NB	94.74	0.8667
rbf-SVM	94.86	0.8617
linear-SVM	97.76	0.9470
DT	97.63	0.9443
Bagging-DT	97.01	0.9249
Ensemble-1	96.91	0.9229
Ensemble-2	98.53	0.9662

It is seen that the classifiers based on the linear regression model created according to the results obtained from the datasets generally give good results. It is also shown that in permission-based malware detection, data in the same class will belong to the same linear subspace and can be expressed by a linear equation.

Since there is a linear relationship between the dataset and the samples, it is possible to make predictions for other samples through the linear regression technique. Finally, it should not be ignored that the obtained bagging techniques also give good results. In the creation of bagging techniques, since the datasets are relatively small, the training parts of the datasets are randomly divided into five parts. It is possible to obtain higher performances by creating more subsets in larger datasets. Also, in this study, different regression models are created by assigning random values to the regression coefficients. Findings of randomly generated models are included in Remark 1.

Remark 1: The regression coefficients obtained in this study generally vary between −1 and 1. 10000 regression models are created by giving random values between −1 and 1 to the regression coefficients. However, the error rates of random models are higher than the actual model. For example, in experiments on the AMD dataset, the Pearson correlation coefficient of the actual regression model is 0.8836. The result of the best randomly generated model is 0.8694 according to the Pearson correlation coefficient. Only 3429 of these random models have Pearson correlation coefficient above 0.80. Better models can be created by developing smart search strategies instead of brute-force searching.

COMPARISON WITH PREVIOUS WORKS:

In this subsection, the results obtained will be compared with some results in the literature. Table 8 compares the results of existing studies with the results obtained in this study. While making comparisons, not only static analysis is taken into account, but also the results obtained from some dynamic and hybrid studies are included.

Comparisons are made with the highest performances reported in existing studies and the classification algorithms in which these performances are obtained.

In this study, since a authorization- grounded Android malware discovery system is proposed, authorization-grounded models will be estimated among themselves first. A general comparison will also be made. According to Table 8, there are 5 studies that only use permission as an attribute. The loftiest performance attained from these studies is attained from the AndroAnalyzer as0.9820 according to the f- measure metric. Using the same dataset, the result of0.9662 is attained according to the f- measure metric with the Ensemble- 2 fashion. Our result is roughly 2 lower than. still, the computational cost of the DNN fashion is relatively high. In addition, the creation of the network is relatively complex as there are numerous parameters. A distribution analogous to this dataset is used.

CONCLUSION AND FUTURE WORKS:

Application permissions are significant in Android operating system security. These permissions, which are extracted from applications, are used as attributes to detect malicious software with machine learning algorithms in this study. Android malware detection is carried out with two rule-based classification models using multiple linear regression models. The proposed rule-based classifiers are compared with popular classification algorithms such as KNN, NB, SVM, and DT.

Both approaches give more successful results than NB and KNN. There are many parameters in SVM, KNN, and NB algorithms. However, classifiers based on multiple linear regression models are quite simple and easy to use. This is the most significant advantage of the proposed approaches. In addition, ensemble learning models based on the bagging technique are also developed in this study. The use of these models positively affects classification performance in general. Finally, in the multiple linear regression model, a large number of models are created by assigning random values to the regression coefficients.

However, positive results cannot be obtained from these models. In future studies, it is aimed to create more efficient regression models by developing intelligent search strategies such as hybrid or heuristic techniques.

ACKNOWLEDGMENT:

The authors would like to express their gratitude to the anonymous reviewers for their invaluable suggestions in putting the present study into its final form.

REFERENCES:

[1] (2021). Global Market Share Held by the Leading Smartphone Operating Systems in Sales to End Users From 1st Quarter 2009 to 2nd Quarter 2018. Accessed: Oct. 30, 2021. [Online]. Available: https://www. statista.com/statistics/266136/global-market-share-held-by-smartphone operating-systems/

[2] (2021). Malware Disguised as Minecraft Mods on Google Play— Kaspersky Official Blog. Accessed: Oct. 30, 2021. [Online]. Available: https://www.kaspersky.com/blog/minecraft-mod-adware-google-play revisited/40202/