INTERNATIONAL RESEARCH JOURNAL OF HUMANITIES AND INTERDISCIPLINARY STUDIES
(Peer-reviewed, Refereed, Indexed & Open Access Journal)
DOI : 03.2021-11278686 ISSN : 2582-8568 IMPACT FACTOR : 5.71 (SJIF 2021)

MALICIOUS ANDROID APPS IDENTIFICATION APPLYING MACHINE LEARNING TECHNIQUES

G. Mahalakshmi[1], R. Amirthalingam[3], B. Senthilnayaki[3], J. Duraimurugan[4], N. Anbarasi[5]

Department of Information Science and Technology, College of Engineering Guindy, Anna University Chennai[1]

Computer Centre, Madras Institute of Technology, Anna University Chennai[2]

Department of Information Science and Technology, College of Engineering Guindy, Anna University Chennai[3]

Department of Information Science and Technology, College of Engineering Guindy, Anna University Chennai[4]

Department of Information Science and Technology, College of Engineering Guindy, Anna University Chennai[5]

DOI No. 03.2021-11278686 **DOI Link ::** https://doi-ds.org/doilink/09.2021-29315818/IRJHIS2109016

ABSTRACT:

The attraction of users towards Android mobile devices is increasing day by day due to its ubiquitous characteristic like easy accessing of information from anywhere and anytime. The availability of open app market systems, leads to the cause for the escalation of malicious Android apps. The increasing usage of mobile device leads to development of new applications, which ends up with more number of application creation for all day to day transactions, other activities and entertainment purposes. The complexity and variety of the vindictive Android applications render the traditional malware recognition strategies ineffectual, which brings about an enormous number of malevolent applications staying undetected. For this reason it is necessary to have an irresistible technique for identification and classification of Android malicious applications. Hence, in this proposed work, a mechanism is presented, called malign android application detection system using Decision tree algorithm and Support Vector Machine. Since the launch of the smartphones and it has become an important part of our lives. Now a day's peoples are very much dependent on smartphones applications for their entertainment reasons, the huge several of applications are downloaded by the users from play store app or from the trusted third party applications. Also due to the unawareness of users, the applications may be downloaded from unauthorized sources pose a threat as it doesn't undergoes the necessary checks or mechanisms to validate the authenticity of these applications and maybe infected with malware. The malware injected applications creates a hole to the flow of user's personal information or to obtain unauthorized access to the system. In this work, we proposed an auxiliary mechanism using C-S DTFS algorithm for malicious app identification by using machine learning techniques for classification of the applications as either malicious or benign and the obtained results are compared to identify the best suited algorithm for our dataset. The issues in existing work are that they used Random forest and Naive Bayes Algorithm. The main issue of random forest is that if number of trees are more in number, will lead to the too slow execution of algorithms results in unsuccessful predictions.

Keywords: *C-S DTFS, Information Gain, Support Vector Machine, Decision Tree*

I. INTRODUCTION:

In the past few years, handheld devices, like Smartphone's, tablets, etc. have become prominent by raising the number and complexity of their capabilities. Present mobile devices offer a large number of services and applications as compared to the one's offered by personal computers. Malwares are specially created for damaging and stealing the user information through the android applications [1]. During the software installation process, the users will ask for permissions to access the photos, audio, contacts and other information. Once the user-approved the permissions, unknowingly the information are stole by the malicious applications [2]. This increases the

total of security threats to the handheld devices. Hence the hackers and malicious users are taking granted of causing threat to the user's personal credentials due to the lack of security mechanisms. In 2020, malware attacks increased to a count 3,246,284 malware samples. Android software reaches the highest malware growth rate by the end of 2020. So it is necessary to use Malware detection for android and can avoid Malware attacking our Android devices when we download from Unknown sources.

Machine Learning problems can be solved effectively and efficiently once the problem has been fully understood. In the proposed work, we explore the classification process for detecting apps as malware or non malicious. This process is called Spam Detection, which is type of binary classification problem. In the existing system, several approaches are introduced in order to address the issues of permission systems. The permissions are categorized into four types: normal, dangerous, signature, and signature Or System. In this the first type it doesn't create harm to the user; because the permissions are granted automatically without the need of user. Whereas in second type, it create negative impacts on incorrect usage. On the other one (dangerous) the user may grants the requested permissions because many of the users are not aware of the security issues that what the permission are actually means, allowing the application to access the user's sensitive information.

In the proposed work, C-SDTFS algorithm is used for effective and efficient feature selection in order to reduce false positive rate which greatly impacts on classification process. The efficient feature selection improves the classification result thereby achieving higher accuracy.

This chapter is organized into 7 sections, describing each part of the proposed system with detailed illustration and system design diagrams. The sections are as follows: **Section 2**: This module consists of Literature survey details of the work alongside their detailed methodologies, advantages, disadvantages etc.

Section 3: This module consists of the proposed system design with its preliminary design such as overall Architecture diagram and process flow diagram which tells about the modules integration. **Section 4**: This module consists of detailed system design or module description with their input and algorithmic steps involved in each module to derive the output as per the user requirement. **Section 5**: This module consists of the details about the hardware and software requirement for the implementation of the proposed system and the experiments that has been performed along with their outcomes. The detailed result of the proposed system is also portrayed in this chapter. **Section 6**: This module concludes the proposed work report with all the results and an implementation procedure that has been underwent during the system development. The future works and excellence of implemented proposed system is detailed.

II. LITERATURE SURVEY:

Naser Peiravian et.al In this paper, author proposes a scheme for combining API calls along with permissions. Machine learning based methods are applied to detect the malware or malicious android applications. In the proposed design, the permissions are actual ally extracted from all individual App's profile information. By using the packages and classes which represent the API calls, the required APIs are extracted. With the help of features like permissions and API calls, the apps are characterized, with this it is possible to learn a classifier to identify App as malicious or not [1].

U.S. Jannat et.al [2] in this paper author presents a various studies on analyses of different android applications. There are two major methods called static and dynamic with which the Android mobile applications are analyzed in order to segregate the harmful applications from the benign ones. This paper shows the various studies and presented the analyses of machine learning models which are used in many android mobile applications. Various features are considered for analyses using different classifiers, the results shows the dynamic analysis model that reach the accuracy of 93. In, this author focuses on one kind of applications. The analyses are not done with different types of applications. H. Soni et.al in the proposed work, author applies Machine Learning approaches for differentiating the genuine and malware android applications [3]. The previous existing datasets of malevolent applications has been obtained with the aid of Help vector machine calculation and the calculation of choice tree which make up relationship with preparing dataset. This dataset as very much helps to predict the malware android applications of up to 93.2. L.wei et al [4] the author proposes a technique using machine learning-based approach for detecting malicious malware applications. In this the author uses a technique to implement malicious application detection tool called Android detect. The first step in this process is to identify the dependency between the system functionalities, protected permissions, and valuable android mobile application programming interfaces. The combined system functions are used to describe the behaviors of the application and also to construct eigenvectors. The values of eigenvector are used to compare the different types of methodologies like naive bayesian, J48 decision tree, and application functions of decision algorithm for effective malware application identification. Fathima et al [5] has proposed an effective machine-learning based approach along with Genetic algorithm for discriminatory feature selection for identification of malware applications. Features that are used for training a system for classification are obtained from Genetic algorithm. The results of proposed system validate that Genetic algorithm gives most optimized feature subset helping in reduction of feature dimension to less than half of the original feature-set. H. Han et.al [6] a framework for malicious android app detection has been proposed, which works based on machine learning algorithm. The proposed framework has designed in such a way to provide high level security also better privacy on user's information. The machine learning based classification algorithm is used to segregate the malicious and genuine applications by monitoring the several permissions of features and events which are obtained from android applications. Luis Ariosto Serna Cardona et al [7] in this paper author proposed an approach using traditional classifiers for the identification of categorical variables (LDC-QDC-KNN-SVM) also various feature mapping techniques for the purpose of increasing the separation of classes. Here the basic features so called (categorical attributes) are mapped with another sample space by using Chi-Square (C-S) algorithm for the measuring of independent features or dissimilarity between the features. Also, the (t-SNE) technique is used in order to reduce the dimensionality of the data, where the computational times for learning methods is also greatly reduced.

Jaemin Jung [8] in this paper, the author proposed an API based malicious application identification technique for detecting the sample space Android APIs for classifying the features. This proposed technique builds a two Android API ranked lists(benign APIlist and malicious APIlist). The benign APIlist holds the frequently called APIs and malicious APIlist holds frequently called API. For a malicious app, it is necessary to compute the ranking value for both the benign API and malicious API using sum of inverse values. The decision of benign or malicious app is identified by determining the comparison values of two sums. Since the achieved accuracy is around 89.93% which is less when compare to other algorithms. The accuracy needs to be improved. Ebtesam J. Alqahtani et al [9], the author provides a survey on the recent techniques for identification of malicious android mobile application which mainly interested in Machine Learning-based classification algorithm for this purpose. Comparing with many existing available approaches and techniques for detecting malicious apps, machine learning-based approach is considered more suitable for detections because of obtaining a high accuracy in detecting the malwares. Hence, it is necessary for the Android devices to have a capability of detecting malwares with the help of machine learning based approaches.

Liu and .XLiu. [10] In this proposed work, author developed a scheme called two-layered permission based detection, in order to detect a malicious Android mobile applications. The study on existing researches shows that, the consideration of using permissions in order to enhance the accuracy of detection. The evaluation results show that, a two-layered permission-based detector has high detection rate of malware.

Wei Wang et. al [11] , in this paper, author provides the survey on techniques that are used to identify the features for effective detection of malicious apps. This survey focuses on four perspectives (extracted features, method for feature selection, methods used for detection and level of performance). It is identified that dynamic analysis provides better results for the apps having powerful functions. But the dynamic analysis is considered to be a more time consuming process when the number of features are more. The machine learning is considered suitable for the apps that have more number of features and deep learning is suitable for intelligent decision making process.

Anıl Utku et. al [12] in this proposed work, android application malware identification system is developed using decision tree algorithm, C4.5 and Hoeffding tree algorithms. The success rate of the C4.5 decision tree algorithm was 95.862 percent and the success rate of the Hoeffding tree algorithm was 93.187 percent. Enriching mobile devices, fast and intensive provides information sharing. However, mobile the increase in the use of devices is also malignant It also causes a serious increase in the number of software. Mobile malware mostly involves the Android platform targets. The reason for this situation is Android system has a large share of the mobile device market and allowing malware to be easily distributed. Research on mobile malware, Malware on the Android platform is constantly show that it has increased. New on average every 14 seconds In researches where a malware has emerged, 560,671 new ones emerged in the second quarter of 2015 Android malware compared to the first quarter of the same year It has been reported to show a 27 percent increase . Hanqing Zhang et al [13]. In this paper, author proposes a novel Android malware detection method based on the method-level correlation relationship of application's abstracted API calls. First, we split each Android application's source code into separate function methods and just keep the abstracted API calls of them to form a set of abstracted API calls transactions. And then, we calculate the confidence of association rules between the abstracted API calls, which forms behavioral semantics to describe an application. Finally, we combine machine learning to identify the different behavioral patterns of malicious and benign apps to build the detection system. The results of evaluation show that the proposed system is competitive in terms of classification accuracy and detection efficiency. At dataset Drebin (benign 5.9K and malware 5.6K) and AMD (benign 20.5K and malware 20.8K), our system has achieved 96 percent and 98 percent detection results both in accuracy and F-measure. Natarajan et al. [14] In this paper, author investigated on basic parameters such as accuracy, precision and recall over two feature selection algorithms namely Chi-Square feature selection and Boruta feature selection algorithms. Observations of the experiments conducted using R studio resulted around 5–6performance in above said parameters when they were exposed to Boruta feature selection algorithm. The experiment was done on two different datasets with different set of features. Boyuan Peng et.al.[15] In this paper author proposed a new effective feature extraction method for efficient android malware identification and detection without affecting user privacy. In this, author focused on different aspects for extracting malware features. Also, XGBoos, LGBM and Enhanced TextCNN models are used to train the sets for effective prediction. At the end these models are integrated with the Stacking model for obtaining the final output. Zibin Guan et al. [16] in this paper, RNN-based systems are analysed for malicious web code detection. By applying various methods like LSTM, GRU, SRU, minimalRNN, it is recommended that textCNN has produced effective detection result when compared to other methods. Jihyeon Park et.al [17] in this paper, author proposed an effective malicious detection method mainly designed for complex operation applications or heavy usage of applications. Here Random Forest classifier is used to classify the genuine and malicious applications based on the features like build-in permissions and custom permissions which are chosen from manifest.xml file. The relationships between N-permissions are analyzed from a subset of permissions. In this the author concluded that the features which are frequently requested permissions as provide the best result in detection of malicious apps. Seif EIDein Mohamed et. al[18] in this author proposes a mechanism using machine learning based techniques for analyzing the features, extracting the features and decision making process for detecting the malicious apps. Malgenome and Maldroid data-sets are used. The machine learning algorithms called KNN, SVM, Naïve Bayes and Decision Tree classifiers based model are applied to the above datasets. The result shows that all the algorithms are achieving great accuracy when compared to Naïve Bayes algorithm. So, the result of Naïve Bayes algorithms is not utilized for decision making process. Between the two datasets Malgenome dataset has achieved higher accuracy of 99.9% on linear kernel with SVM classifier. Jatin Acharya et.al [19] has proposed a solution for virus and Trojan attacked android applications, which had great challenge in detecting these kinds of applications. Machine Learning and Signature matching techniques are used to provide a solution for this detection problem. The Random Forest model is trained using features contains in Portable Executable file headers. For detecting the virus affected applications, using Signature matching technique to match the MD5 hash of the file with signature database. Logistic Regression model is used to differentiate malicious and genuine URLs. The TfidfVectorizer is used for converting the text value into weighted value. The proposed model has f1-score value of 0.89 for malicious URL which needs to be improved by adding more values. Sivasangari et.al [20] author has formulate a adaboost formula for the problem of detection of SQL injection attack which has major impact on internet security. The weak tree is given a higher weight for getting the strong model thereby updating the weights during each step of dataset training process. It takes high time in detecting the data in real time projects. In the literature survey, authors are focused on identifying the android app as either malicious or genuine apps, instead not focusing on considering the attacks that are related to the features. The features are need to categorized in order to identify to which attack it is mainly related with.

III. ARCHITECTURE DESIGN:

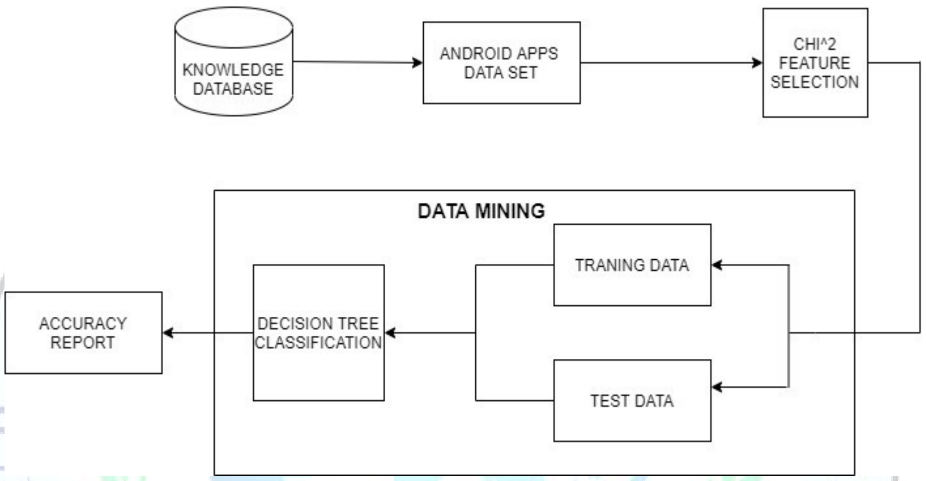


Figure 3.1: Proposed Android Malicious App Detection System Design

The target is to find the malicious apps using the given data set by using the tree based Machine Learning Algorithm that is Decision Tree algorithm. Decision Tree classifiers algorithm applied on dataset so as to detecting Malicious Android Application.

3.1 DESCRIPTION ABOUT THE DATASET:

It is challenging task to accumulate data related to android application, hence, the 'DREBIN' dataset is used which contains 15036 data's of which 123,453are genuine android applications, the remaining 5,560 android applications contain malicious sample space which belongs to 179 various malware groups and other 9476 are benign samples. The highest ranking 20 groups of malware are shown below Table 3.1.

Table 3.1: Important malware groups in dataset

Malware family	# Entities	Malware family	# Entities
FakeInstaller	925	Adrd	91
DroidKungFu	667	DroidDream	81
Plankton	625	ExploitLinuxLotoor	70
Opfake	613	Goldream	69
Ginmaster	339	MobileTx	69
BaseBridge	330	FakeRun	61
Iconosys	152	SendPay	59
Kwin	147	Gappusin	58
fakeDoc	132	Imlog	43
Geinimi	92	SMSreg	41

The data consists of 215 features where it has been categorized into eight types and the dataset is made up of benevolent apps and the malware affected apps; where each file or app contains necessary features like hardware components, requested permissions, app components, network addresses, etc. The Main Features are explained below.

Hardware Components : The required hardware component set.

For example, touchscreen, camera, GPS and so on.

Requested Permissions: These are the permissions that are requested during app installation process in order to ensure security. The user needs to provide or deny the permissions based on the nature of the application. This will allow the user to deny or restrict the permissions for the case of malicious applications.

Application Components: Application components are the essential building blocks of an Android application. These components are loosely coupled by the application manifest file AndroidManifest.xml that describes each component of the application and how they interact. There are following four main components that can be used within an Android application Activities, Services, Broadcast Receivers, Content Providers.

Filtered intents: Intents performing a action of inter and intra process communication.

For example, BOOT_COMPLETED is widely used in order to trigger the malicious activities directly once the system has been rebooted.

Restricted API calls: This API calls is work depend upon the permissions that are accepted during app installation process. There may be some cases where the permissions may not be asked during installation.

Suspicious API calls: In some cases, there may be a chance for obtaining access to sensitive device information's through API calls, which can be used for obfuscation.

Network addresses : These are the addresses that are used by malicious applications for the purpose of sharing data and also to execute the external commands.

Flow Diagram : The flow diagram shows the flow of process for detecting the malicious android apps.

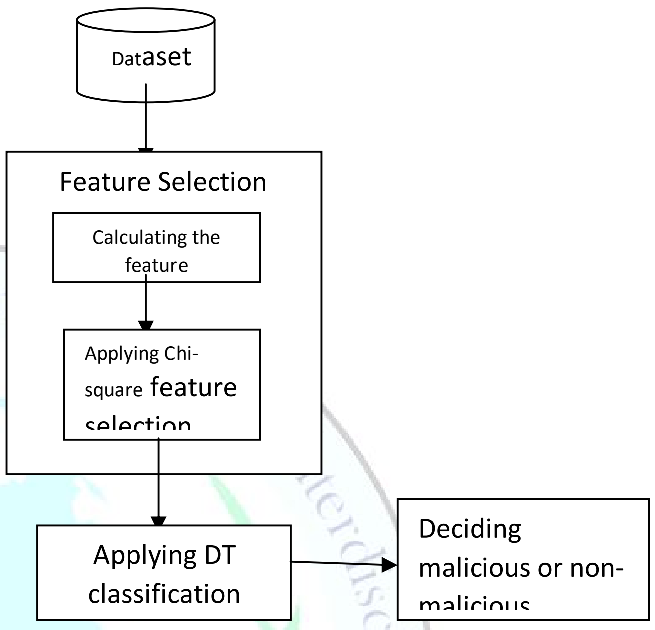

Dataset
Feature Selection
Calculating the feature
Applying Chi-square feature selection
Applying DT classification
Deciding malicious or non-malicious

Figure 3.2 Flow diagram for proposed system

IV. PROPOSED ALGORITHM:

This section gives the detailed description about the different modules of proposed system. Each module contains input, process flow and output for the module.

4.1 Chi Square Based Feature Selection:

The Chi-Square test is performed to determine, the relationship between the two categorical variables. The Chi-Square statistic is calculated as follows:

$$x = \Sigma \frac{(Observed-Expected)2}{Expected} \qquad (3.1)$$

$$\text{Expected} = \frac{RowTotal*ColumnTotal)}{OverallTotal} \quad (3.2)$$

4.2 Information Gain:

Information gain is a measure that is used to calculate the reduction value in entropy after reconstructing the dataset. The remodeling of dataset is based on the attributes.

This information gain will calculate the percentage of information that each provides about the class data. Based on the value of information gain the decision tree nodes will be constructed. The decision nodes with maximum information gain value will be splitted first. The formula for calculating the information gain is:

$$Entropy(s) = -P(yes)\log_2 P(yes) - P(no)\log_2 P(no) \qquad (3.4)$$

4.3 Gini Index

Gini index is a measure of impurity or purity used while developing a Decision Tree algorithm. An attribute having low Gini index value is usually preferred when compared to the attribute having high Gini index. Gini index can be calculated using the below formula.

$$GiniIndex = 1 - \sum jPj^2 \qquad (3.5)$$

4.4 Support Vector Machine Algorithm:

Support Vector Machines (SVM) comes under the category of supervised learning methods which is normally used for classification and regression problems. The goal of SVM is to identify an optimal separating hyperplane which maximizes the margin between different classes of the training data. The advantages of SVM include that they can be used to avoid the difficulties of using linear functions in the high-dimensional feature space. SVM is more effective in high dimensional spaces. SVM is effective in cases where the number of dimensions is greater than the number of samples. SVM is relatively memory efficient.SVM problem can be formulated as,

$$w.x_i + b \geq 1 \quad \text{for} \quad y_i = +1 \qquad (3.6)$$

$$w.x_i + b \leq -1 \quad \text{for} \quad y_i = -1 \qquad (3.7)$$

4.5 DATA COLLECTION;

The data consists of 215 features where it has been categorized into eight classes. The dataset contains genuine apps and malicious apps.

INPUT : Data collecting as CSV Data sets.

PROCESS : Data Used is separated as Independent and Dependent data and Used for Further Processes like Feature Selection and Classification.

Steps For Data Collection

1: Go to derbin dataset website
2: Request for the dataset from the owner
3: Fill the details of the application.
4: Wait for Authorization From owner.
5: The data set will be sent to u as zip file by owner.
6: Download the file.
7: Extract the Zip file which u recieved.
8: store as a CSV File.

OUTPUT: Maintain the whole raw data as a csv file for further process.

4.6 FEATURE SELECTION:

The Chi-Square test is a type of statistical test, is perform to determine relationship between 2 categorical variables. In other words, the Chi-Square statistic will find significant deviation value between the observed value and expected value of the variables. Using following steps the Chi-Square statistic is calculated.

INPUT : Related datasets as a csv file.

PROCESS : Chi Square based Feature Selection Algorithm used to select the important features which we need to use for the Classification Algorithm to Predict the Acccuracy.

OUTPUT : Most important feature will alone be selected from dataset and will be used for further process.

4.7 C-S DTFS ALGORITHM:

The D-Tree algorithm is one of the supervised learning methodology which can be applied in any of the classification and Regression problems, but this algorithm is mainly used to solve the Classification based problems. As the name implies, this decision tree is a tree-structured classifier, where the nodes of the tree represents the features that are present in the dataset, and the decision making rules are represented by branches and the outcome of the rule is represented by the leaf node. Normally, the Decision tree has two nodes, called Decision Node and Leaf Node. Decision making process is done by Decision node, which contains multiple branches, and the output process is represented by Leaf nodes which does not have any further branches.

INPUT: Final Data set as a csv file.

PROCESS: This models classifies the categorical class labels, and classify the problems.

C-SDTFS Algorithm

```
1: Start constructing the tree with root node, say S.
2: Read data from a database.
3: Find the dependent attribute in the dataset.
4: Partition the S into number of subsets which contains possible values for the best
attributes.
5: Create the decision tree node for the each best attribute.
6: Repeat the steps to create new decision trees using the generated subsets of
the dataset which is created in step 4. Repeat the process until no more
nodes to classify.
```

OUTPUT: Find to Accuracy, Precision, Recall and F1 Scores of algorithms.

4.8 SUPPORT VECTOR MACHINE:

Support Vector Machines (SVM) is a type of supervised learning methods which is used for classification and regression tasks that are originated from statistical learning theory. Similar to classification method, SVM is a global classification model which is used to generate non-overlapping partitions of all attributes. The main use of SVM is for classification. We did plotting on n-dimensional space. The value of feature is considered as the value of the specific coordinate. Then, the ideal hyper plane is identified to differentiate two classes. The working principle of SVM is the create a hyper lane in order to separate the dataset in to different classes. Simply SVM converts the unseperatable problem in seperateable problem.

INPUT : Final Data set as a .CSV file

Algorithm

```
1: Import the necessary libraries for the implementation of SVM .
2: Input the Derbin dataset.
3: Apply split() function to distribute the dataset into training test
   and test set
4: Apply scaling operation on data to verify the data values are lies
   in a common range without extreme values.
5: Apply SVC function using the sklearn library.
6: we get accuracy report.
7: Take necessary actions based on accuracy value.
```

OUTPUT : Find to Accuracy, Precision, Recall and F1 Scores of algorithms

V. EXPERIMENTAL RESULTS

In experimental results consists of the details about the hardware and software requirements to the project and the experiments that have been performed along with their outcomes.

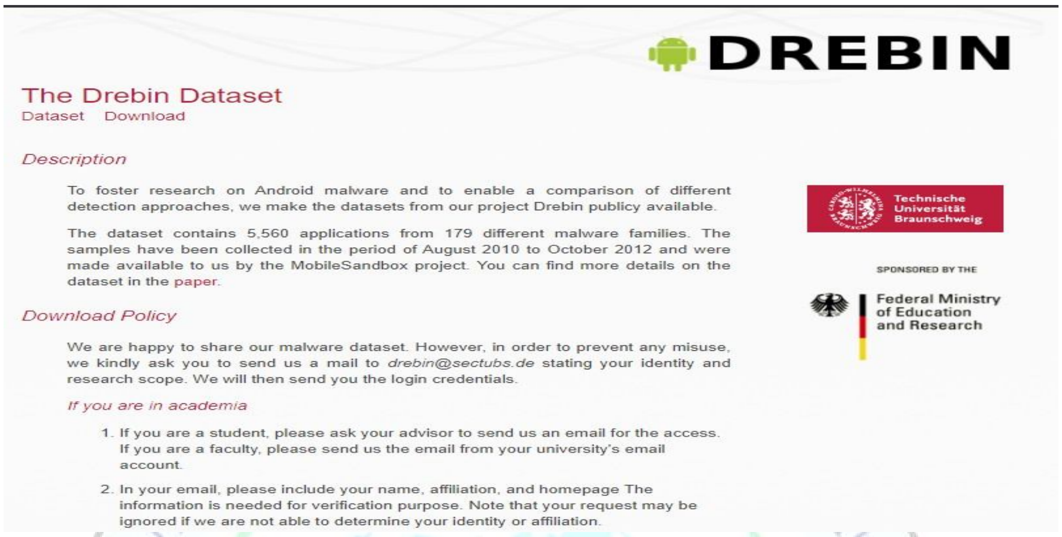
5.1 ANDROID APPLICATION DREBIN DATASET:

The dataset contains of about 215 features, which are categorized into eight types. The dataset consists of genuine apps and malware infected apps. The files or applications contain the feature that includes important hardware components, demand permissions, application needed components, network addresses and so on. The important Features are explained below.

Table 3.2 Prominent features in dataset

Sl. No	Important Features
1.	Hardware Components
2.	Requested Permissions
3.	App Components
4.	Filtered Intents

5.	Restricted API Calls
6.	Used Permissions
7.	Suspicious API Calls
8.	Network Addresses


DREBIN
Technische Universität Braunschweig
SPONSORED BY THE
Federal Ministry of Education and Research

The Drebin Dataset

Dataset Download

Description

To foster research on Android malware and to enable a comparison of different detection approaches, we make the datasets from our project Drebin publicly available.

The dataset contains 5,560 applications from 179 different malware families. The samples have been collected in the period of August 2010 to October 2012 and were made available to us by the MobileSandbox project. You can find more details on the dataset in the paper.

Download Policy

We are happy to share our malware dataset. However, in order to prevent any misuse, we kindly ask you to send us a mail to *drebin@sectubs.de* stating your identity and research scope. We will then send you the login credentials.

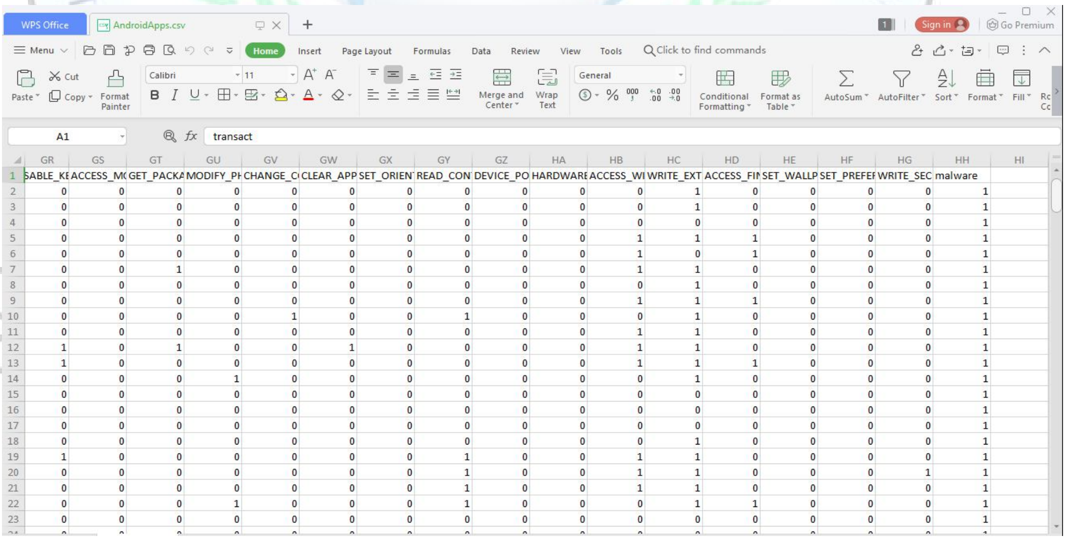
If you are in academia

1. If you are a student, please ask your advisor to send us an email for the access. If you are a faculty, please send us the email from your university's email account.
2. In your email, please include your name, affiliation, and homepage The information is needed for verification purpose. Note that your request may be ignored if we are not able to determine your identity or affiliation.

Figure 5.1: How to get Dataset

5.2 DATASET COLLECTION

The figure 5.2 shows the sample dataset in the csv file format, the file contains 216 independent data and 1 dependent data. The 5,560 applications are considered as malware samples in which are 179 different malware groups and 9476 are benign sample data.

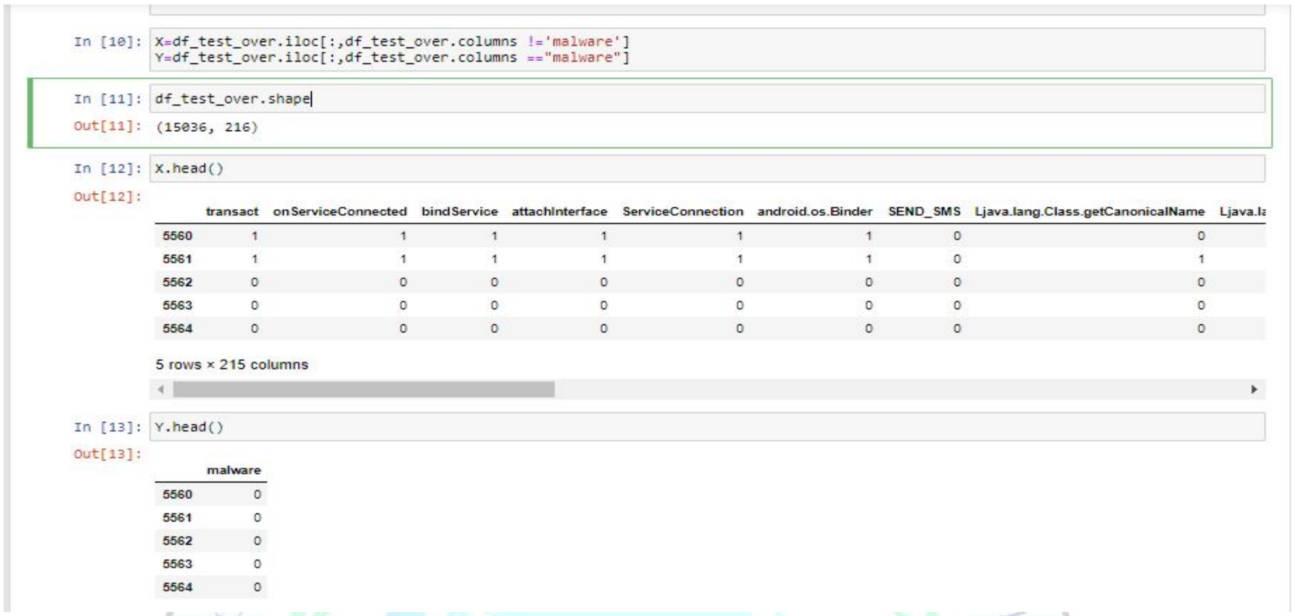


	GR	GS	GT	GU	GV	GW	GX	GY	GZ	HA	HB	HC	HD	HE	HF	HG	HH	HI
1	SABLE_KE	ACCESS_M(	GET_PACKA	MODIFY_PH	CHANGE_C	CLEAR_APP	SET_ORIENT	READ_CON	DEVICE_PO	HARDWARE	ACCESS_WI	WRITE_EXT	ACCESS_FIN	SET_WALLP	SET_PREFEF	WRITE_SEC	malware	
2	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	
3	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
5	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	1	
6	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	
7	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	
8	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	
9	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	1	
10	0	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	1	
11	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	
12	1	0	1	0	0	1	0	0	0	0	1	1	0	0	0	0	1	
13	1	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	1	
14	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	1	
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
18	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	
19	1	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	1	
20	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	1	1	
21	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	1	
22	0	0	0	1	0	0	0	1	0	0	0	1	1	0	0	0	1	
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	

Figure 5.2: Datasets (in CSV) Which are Collected In Drebin Website

5.3 FEATURE SELECTION

The Chi-Square test is a type of statistical test which is used to determine the relationship between the two categorical variables. That is it identifies the difference between the observed values with the expected value of the variables. The Categorization of Independent and Dependent variables are shown in the below figure.

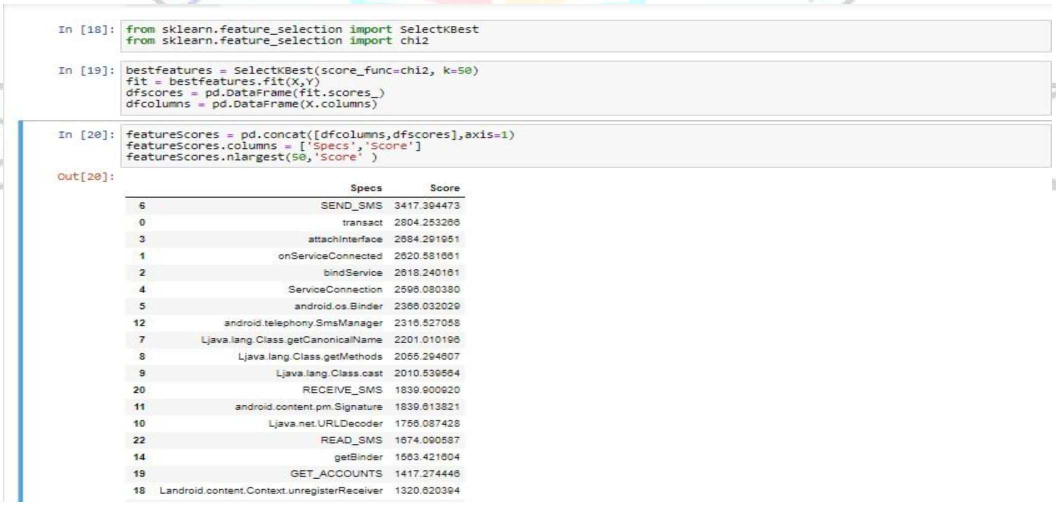


```
In [10]: X=df_test_over.iloc[:,df_test_over.columns !='malware']
         Y=df_test_over.iloc[:,df_test_over.columns =="malware"]

In [11]: df_test_over.shape
Out[11]: (15036, 216)

In [12]: X.head()
Out[12]:
      transact  onServiceConnected  bindService  attachInterface  ServiceConnection  android.os.Binder  SEND_SMS  Ljava.lang.Class.getCanonicalName  Ljava.la
5560  1         1                   1            1                1                  1                  0         0
5561  1         1                   1            1                1                  1                  0         1
5562  0         0                   0            0                0                  0                  0         0
5563  0         0                   0            0                0                  0                  0         0
5564  0         0                   0            0                0                  0                  0         0

5 rows × 215 columns

In [13]: Y.head()
Out[13]:
      malware
5560  0
5561  0
5562  0
5563  0
5564  0
```

Figure 5.3: Categorized Data

When two features are find independent, then the observed value will be more close to the expected value, means the value of Chi-Square is smaller. On the other hand very high Chi-Square value indicates that the incorrect hypothesis independence. Finally it indicates that the high dependency of features when having high Chi-Square value, hence that can be used for training the model.

The Figure 5.4 shows the structured data after selecting the most influential independent features.



```
In [18]: from sklearn.feature_selection import SelectKBest
         from sklearn.feature_selection import chi2

In [19]: bestfeatures = SelectKBest(score_func=chi2, k=50)
         fit = bestfeatures.fit(X,Y)
         dfscores = pd.DataFrame(fit.scores_)
         dfcolumns = pd.DataFrame(X.columns)

In [20]: featureScores = pd.concat([dfcolumns,dfscores],axis=1)
         featureScores.columns = ['Specs','Score']
         featureScores.nlargest(50,'Score' )
Out[20]:
                                           Specs        Score
 6                                      SEND_SMS  3417.394473
 0                                      transact  2804.253266
 3                               attachInterface  2684.291951
 1                            onServiceConnected  2620.581661
 2                                   bindService  2618.240161
 4                             ServiceConnection  2596.080380
 5                             android.os.Binder  2366.032029
12                 android.telephony.SmsManager  2316.527058
 7             Ljava.lang.Class.getCanonicalName  2201.010196
 8                  Ljava.lang.Class.getMethods  2055.294607
 9                       Ljava.lang.Class.cast  2010.539564
20                                  RECEIVE_SMS  1839.900920
11                android.content.pm.Signature  1839.613821
10                       Ljava.net.URLDecoder  1756.087428
22                                     READ_SMS  1674.090587
14                                    getBinder  1563.421604
19                                 GET_ACCOUNTS  1417.274446
18  Landroid.content.Context.unregisterReceiver  1320.620394
```

Figure 5.4: Algorithm for Feature Selection

Feature selections is considered as an important issue in machine learning, because of the presence of several features in line and have to choose the appropriate features for building an efficient and effective model with better visualization of the prominent features as shown in the form of bar graph in Fig 5.5.

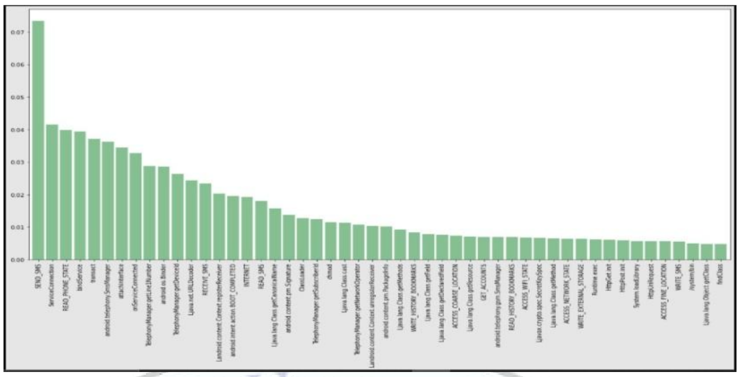


Figure 5.5: Important Features a BarGraph

5.4 EVALUATION METRICS

For the purpose of evaluating the results of the mentioned two algorithms, four important measures are used: Accuracy, Precision, Recall, and F1 score. These four metrics are explained in the following:

1. **Accuracy Rate:** It is a ratio between the number of correctly predicted instances, with the total number of instances.
2. **Precision:** It is a measure which provides ratio between the correctly predicted positive instances with total number of predicted positive instances for each class.

$$P = \frac{TP}{TP+FP} \qquad (5.1)$$

3. **Recall:** It is a measure that provides ratio between the predicted true positive instances and the sum of true positives and false negatives from the observation for each class.

$$R = \frac{TP}{TP+FN} \qquad (5.2)$$

4. **Fl score** provides Precision and Recall with a weighted average also it is considered to be perfect when it is having a value of 1.0 and considered worst with the possible value of 0.0, for a good F1 score it should have low false positives and low false negatives.

$$F = \frac{2PXR}{P+R} \qquad (5.3)$$

5.5 CLASSIFICATION ANALYSIS

The figure 5.6 depicts that the around 5,560 applications are containing malware samples that are belonging from 179 different malware families of which 9476 are benign samples.

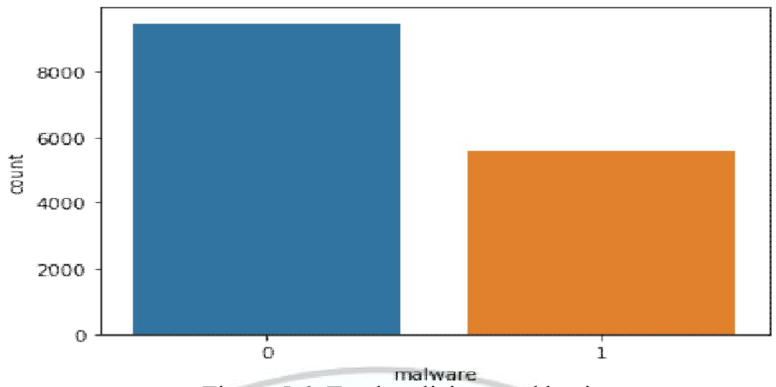
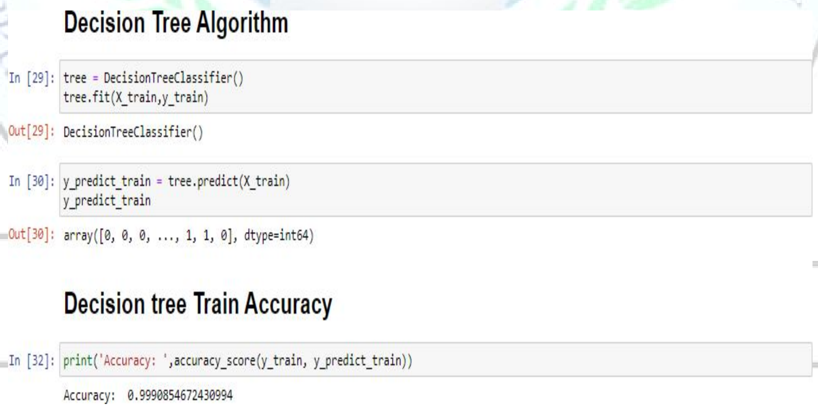


Figure 5.6: Total malicious and benign apps

Decision Tree algorithm is a kind of supervised learning technique which are used for both classification and Regression problems, but generally it is preferred to solve Classification problems. Decision tree represents a tree-structured classifier, in which the internal nodes represent the features of dataset, branches represent the process and each leaf node represents the outcome of the process. The Decision tree is said to contain two nodes called Decision Node and Leaf Node. The Decision nodes are helpful in decision making process that has multiple branches, whereas the Leaf nodes show the output of those decisions which does not contain any more branches. The Figure 5.7 show the Algorithm and result generate by Performing Decision Tree accuracy for Training dataset.



Decision Tree Algorithm

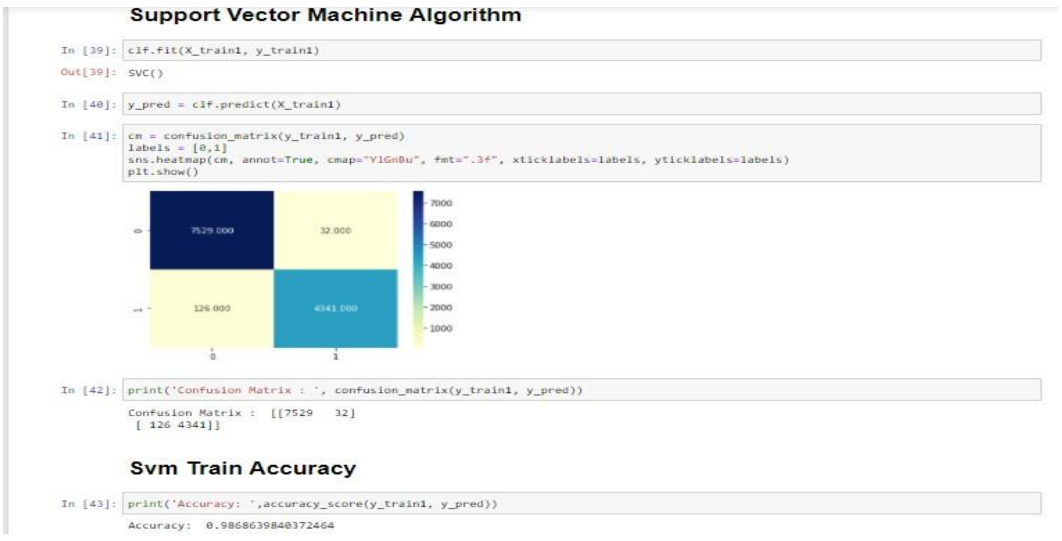
```
In [29]: tree = DecisionTreeClassifier()
         tree.fit(X_train,y_train)

Out[29]: DecisionTreeClassifier()

In [30]: y_predict_train = tree.predict(X_train)
         y_predict_train

Out[30]: array([0, 0, 0, ..., 1, 1, 0], dtype=int64)
```

Decision tree Train Accuracy

```
In [32]: print('Accuracy: ',accuracy_score(y_train, y_predict_train))

         Accuracy:  0.9990854672430994
```

Figure 5.7: Decision Tree Training Results

Support Vector Machines (SVM) is a type of supervised learning method which can be used for classification and regression process, which are originated from statistical learning theory. SVM is a global classification method which generates many non-overlapping partitions of all the attributes.



Support Vector Machine Algorithm

```
In [39]: clf.fit(X_train1, y_train1)
Out[39]: SVC()

In [40]: y_pred = clf.predict(X_train1)

In [41]: cm = confusion_matrix(y_train1, y_pred)
         labels = [0,1]
         sns.heatmap(cm, annot=True, cmap="YlGnBu", fmt=".3f", xticklabels=labels, yticklabels=labels)
         plt.show()
```

```
In [42]: print('Confusion Matrix : ', confusion_matrix(y_train1, y_pred))
         Confusion Matrix :  [[7529   32]
          [ 126 4341]]
```

Svm Train Accuracy

```
In [43]: print('Accuracy: ',accuracy_score(y_train1, y_pred))
         Accuracy:  0.9868639840372464
```

Figure 5.8: SVM Training Results

The figure 5.8 represents the Training results of SVM, the experimental data will be randomly distributed into a training data set and testing data set to detect the two parts.

5.6 COMPARISION OF EVALUATION RESULTS:

The figure 5.9 depicts the accuracy rate of DT based Android Malware detection system which achieved 99.90% in Training data, and the recall rate achieves 98%, and the F- measure achieves 97%, which shows the satisfactory results of proposed algorithm.

Parameters	Accuracy	Precision	Recall	F Measure
DT	99.90	97	98	97
SVM	98.6	96	97	96

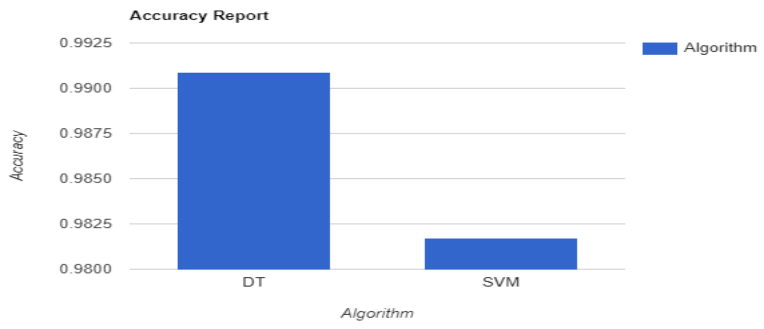
Figure 5.9: Evaluation Training Data Results of Two Algorithms

Compared with the similarity algorithm based on the SVM, the overall accuracy rate is considerably enhanced by 98.6 percent, also there is an increase in the overall of recall rate is about 97 percent, and the average F-measure value is increased by 96 percent for training data.

5.10 RESULT ANALYSIS:

The proposed algorithm achieves higher value compare to other existing algorithms. It is observed that when number of detected tweet texts increasing, then the detection accuracy decreases, and also the recall rate and F measurement increases. The reason for this is due to the increase in detected text, it increased the scope of the query, so resulting in increased recall rate.

Figure 5.10: Comparison Graph of Two Algorithms

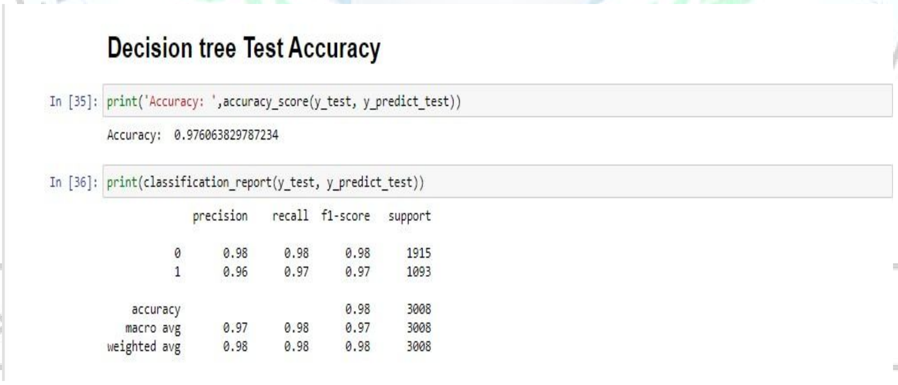


From the Figure 5.10 it is observed that the accuracy rate of similarity algorithm based on DT algorithm achieves higher value than SVM.

5.11 ACCURACY OF DECISION TREE USING TEST DATA:

In Fig.5.11, shows the test accuracy of the decision tree algorithm. A testing dataset is considered has independent of the training dataset, but it follow the same probability distribution similar to the training dataset. Once a model fit in to the training dataset then it will also fits to the testing dataset well, with minimal overfitting. A best fitting of the training dataset as opposed to the testing dataset will usually points to overfitting.

A test set is the one which contains a set of examples which are used for performance assessment of classifier. In order to perform this, we used the last model for predicting classifications examples in the test dataset. These are the predictions that are compared to the examples' true classifications to assess the accuracy of the models. The Test Data Accuracy For Decision Tree Algorithm is 97.6 compared to the Train data Of 99.90



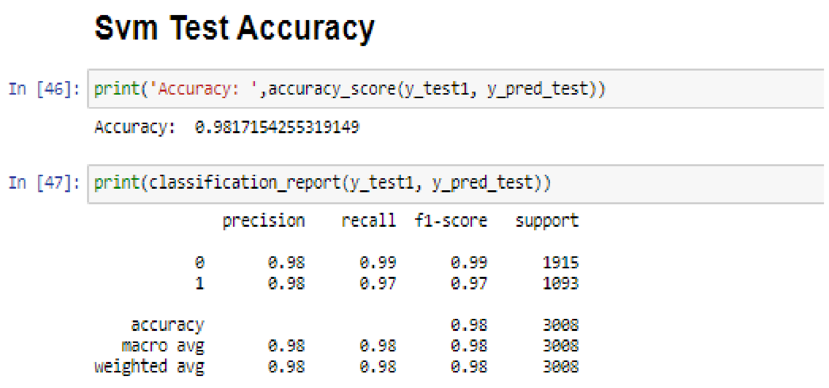
Decision tree Test Accuracy

```
In [35]: print('Accuracy: ',accuracy_score(y_test, y_predict_test))

         Accuracy:  0.976063829787234

In [36]: print(classification_report(y_test, y_predict_test))

                       precision    recall  f1-score   support

                    0       0.98      0.98      0.98      1915
                    1       0.96      0.97      0.97      1093

             accuracy                           0.98      3008
            macro avg       0.97      0.98      0.97      3008
         weighted avg       0.98      0.98      0.98      3008
```

Figure 5.11: Accuracy of DT Algorithm For Test data

ACCURACY OF SUPPORT VECTOR MACHINE USING TEST DATA

The Test Data Accuracy for Decision Tree Algorithm is 98.17 compared to the Train data Of 98.6.



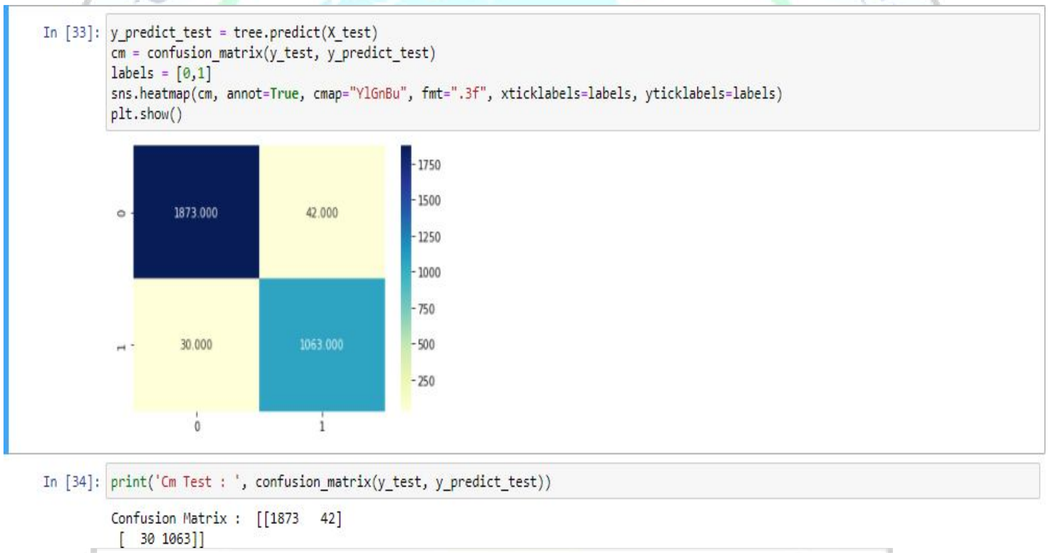
Svm Test Accuracy

```
In [46]: print('Accuracy: ',accuracy_score(y_test1, y_pred_test))

         Accuracy:  0.9817154255319149

In [47]: print(classification_report(y_test1, y_pred_test))

                       precision    recall  f1-score   support

                   0       0.98      0.99      0.99      1915
                   1       0.98      0.97      0.97      1093

            accuracy                           0.98      3008
           macro avg       0.98      0.98      0.98      3008
        weighted avg       0.98      0.98      0.98      3008
```

Figure 5.12: Accuracy of SVM Algorithm For Test Data

CONFUSION MATRIX FOR DECISION TREE

The Fig.5.13 depicts the confusion matrix of Decision Tree Algorithm Results where the result shows that there are 1873 data categorized in TP and 42 data categorized in FP region. In FN region there are 30 data and 1063 data in TN category.



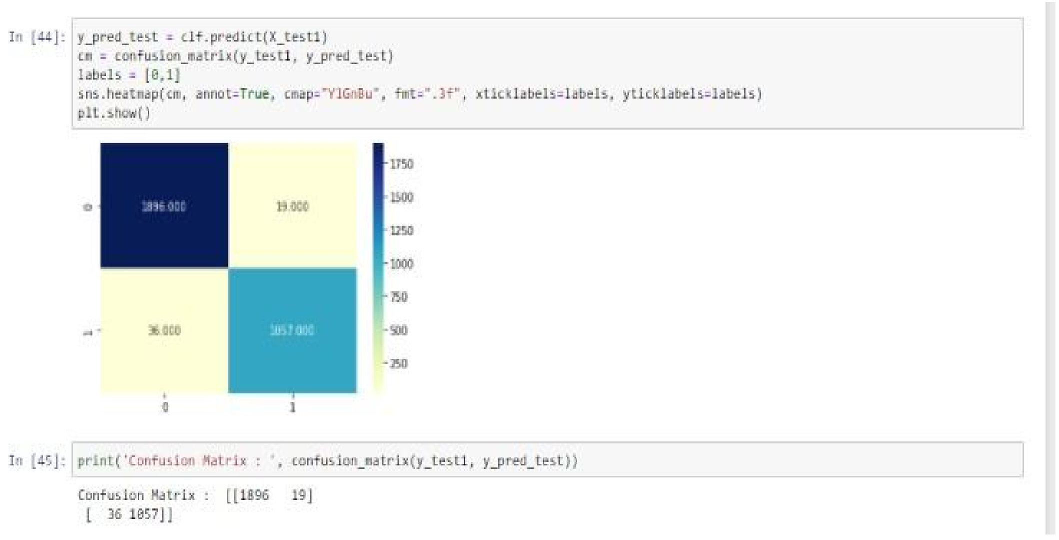
```
In [33]: y_predict_test = tree.predict(X_test)
         cm = confusion_matrix(y_test, y_predict_test)
         labels = [0,1]
         sns.heatmap(cm, annot=True, cmap="YlGnBu", fmt=".3f", xticklabels=labels, yticklabels=labels)
         plt.show()
```

```
In [34]: print('Cm Test : ', confusion_matrix(y_test, y_predict_test))

         Confusion Matrix :  [[1873   42]
          [  30 1063]]
```

Figure 5.13: Confusion Matrix For Decision Tree

CONFUSION MATRIX FOR SVM ALGORITHM

The Fig.5.14 depicts the confusion matrix of Support Vector Machine Algorithm Results where the result shows that there are 1896 data categorized in TP and 19 data categorized in FP region, in FN region There are 36 data and 1057 data in TN category.



```
In [44]: y_pred_test = clf.predict(X_test1)
         cm = confusion_matrix(y_test1, y_pred_test)
         labels = [0,1]
         sns.heatmap(cm, annot=True, cmap="YlGnBu", fmt=".3f", xticklabels=labels, yticklabels=labels)
         plt.show()
```

```
In [45]: print('Confusion Matrix : ', confusion_matrix(y_test1, y_pred_test))

         Confusion Matrix :  [[1896   19]
          [  36 1057]]
```

Figure 5.14: Confusion Matrix For Decision Tree

VI. CONCLUSION:

The functioning of various mobile applications in Android platform was examined and the most eminent malicious activity has been analysed. Also with the dataset, it makes possible reach up to 98% accuracy in labelling the applications as either genuine or malicious. The Cross-Validation is also performed in order to show the strengthness of the proposed algorithm. Aditionally, the Android applications are analyzed to determine the possible malicious activity and its effect. Our study shows that, the Analysis process might indeed perform better to achieve high accuracy using addition features in detecting Android malicious application. The detection system can be further enhanced in future for better accuracy and handling various kinds of new families of virus which or not present in the current. the train accuracy of DT is 99.90% and Test with 97.16% and With SVM train Accuracy was 98.6% And Test accuracy Was 98.17%. in Future We can Expect Maximum Accuracy For the Android Malware Detection System With the Growth Of technology. Future work might be to research and develop a Android App with the enhanced aalgorithm which is installed in mobile system. The app detects the installed application, if found malicious application it notifies and further installation can be stopped that in our android so that we can make our android system more safe and secured.

REFERENCES:

[1] N. Peiravian And X. Zhu. "Machine Learning For Android Malware Detection Using Permission And Api Calls". In 2013 IEEE 25th International Conference On Tools With Artificial Intelligence, Pages 300–305, 2013.

[2] U. S. Jannat, S. M. Hasnayeen, M. K. Bashar Shuhan, And M. S. Ferdous. "Analysis And Detection Of Malware In Android Applications Using Machine Learning". In 2019 International Conference On Electrical, Computer And Communication Engineering (Ecce), Pages 1–7, 2019.

[3] H. Soni, P. Arora, And D. Rajeswari. "Malicious Application Detection In Android Using Machine Learning". In 2020 International Conference On Communication And Signal Processing (ICCSP), Pages 0846–0848,2020.

[4] L. Wei, W. Luo, J. Weng, Y. Zhong, X. Zhang, And Z. Yan. "Machine Learning-Based Malicious Application Detection of Android". IEEE Access, 5:25591–25601, 2017.

[5] A. Fatima, R. Maurya, M. K. Dutta, R. Burget, And J. Masek. "Android Malware Detection Using Genetic Algorithm Based Optimized Feature Selection And Machine Learning". In 2019 42nd International Conference On Telecommunications And Signal Processing (Tsp), Pages 220–223, 2019.

[6] H. Han, S. Lim, K. Suh, S. Park, S. Cho, And M. Park. "Enhanced Android Malware Detection: An Svm-Based Machine Learning Approach". In 2020 Ieee International Conference On Big Data And Smart Computing (Bigcomp), Pages 75–81, 2020.

[7] Luis Ariosto Serna Cardona, Hern ÁN DarIo Vargas-Cardona, Piedad Navarro GonzáLez, David Augusto Cardenas PeñA, And ÁLvaro ÁNgel Orozco GutiéRrez. "Classification Of Categorical Data Based On The Chi-Square Dissimilarity And T-Sne. Computation", 8(4), 2020.

[8] J. Jung, K. Lim, B. Kim, S. Cho, S. Han, And K. Suh. "Detecting Malicious Android Apps Using The Popularity And Relations Of Apis".In 2019 Ieee Second International Conference On Artificial Intelligence And Knowledge Engineering (Aike), Pages 309–312, 2019.

[9] E. J. Alqahtani, R. Zagrouba, And A. Almuhaideb."A Survey On Android Malware Detection Techniques Using Machine Learning Algorithms" . In 2019 Sixth International Conference On Software Defined Systems (Sds), Pages 110–117, 2019.

[10] X. Liu And J. Liu."A Two-Layered Permission-Based Android Malware Detection Scheme". In 2014 2nd Ieee International Conference On Mobile Cloud Computing, Services, And Engineering, Pages 142–148, 2014.

[11] Wei Wang, Meichen Zhao, Zhenzhen Gao, Guanquan Xu, Hequn Xian,Yuanyuan Li and Xiangliang Zhan, "Constructing Features for Detecting Android Malicious Applications: Issues, Taxonomy and Directions", IEEEAccess, Volume 7, 2019.

[12] A.Utku, İ. A. DoğRu, And M.A. Akcayol. "Decision Tree Based Android Malware Detection System". In 2018 26th Signal Processing And Communications Applications Conference (Siu), Pages 1–4, 2018.

[13] H. Zhang, S. Luo, Y. Zhang, And L. Pan. "An Efficient Android Malware Detection System Based On Method-Level Behavioral Semantic Analysis". Volume 7, Pages 69246–69256, 2019.

[14] Bhalaji Natarajan, Sundharakumar Kb, And Chithra Selvaraj." Empirical Study Of Feature Selection Methods Over Classification Algorithms". International Journal Of Intelligent Systems technologies And Applications, 17:98, 01 2018.

[15] Boyuan Peng And Hefei, "Research On Detection Of Malicious Software", 2nd International Conference On E-Commerce And Internet Technology((Ecit), 2021.

[16] Zhibin Guan, Jiajie Wang, Xiaomeng Wang, Wei Xin, Jing Cui And Xiangping Jing, "A Comparative Study Of Rnn-Based Methods For Web Malicious Code Detection", Ieee 6th International Conference On Computer And Communication Systems", Ieee, 2021.

[17] Jihyeon Park, Munyeong Kang, Seong-Je Cho, Hyoil Han And Kyoungwon Suh, "Analysis Of Permission Selection Techniques In Machine Learning-Based Malicious App Detection", Ieee Third International Conference On Artificial Intelligence And Knowledge Engineering(Aike), Ieee 2020.

[18] Seif Eidein Mohamed, Mostafa Ashaf, Amr Ehab, Omar Shereef, Haytham Metwaie And Eslam Amer, "Detecting Malicious Android Application Based On Api Calls And Permissions Using Machine Learning Algorithms", International Mobile, Intelligent, And Ubiquitous Computing Conference (Miucc), Ieee 2021.

[19] Jatin Acharya, Anshul Chaudhary, Anish Chabbria And Smita Jangale, "Detecting Malware, Malicious Urls And Virus Using Machine Learning And Signature Matching", Second International Conference For Emerging Technology (Incet), IEEE 2021.

[20] A.Sivasangari, J.Jyostna And K.Pravalika, " Sql Injection Attack Detection Using Machine Learning Algorithm", Proceedings Of The Fifth International Conference On Trends In Electronics And Informatics(Icoei)", Ieee Xplore 2021.