WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Prediction of Malicious Android Applications Using Machine Learning Approaches

Dr. D. Hema Latha[1], Dr. D. Rama Krishna Reddy[2], Sudha Katkuri[3]

[1] Assistant Professor, Dept of Computer Science, University College for Women, Koti, Hyderabad, TS, India
[2] Assistant .Professor, Dept of Mathematics, Osmania University, Hyderabad, TS, India
[3] Assistant .Professor, Dept of Business Management, RBVRR Women's College, Hyderabad, TS, India

ABSTRACT

Android has become 70% - 80% popular operating system for smart phones and tablets. Along with this popularity, cyber-criminals have expanded their malicious activities to mobile platforms. Mobile threat Researchers have recognized an alarming increase in threat for Android malware from 2012 to 2013 and estimated the number of malicious applications detected is in the range of 120,000 to 718,000. Many efforts have been contributed to study the nature of smart phone platforms and their applications, to detect malware from work place and third-party sources applications efficiently.*The ultimate goal of this work is to improve permission to detect malicious android mobile applications using machine learning approaches*.First, dataset of previous malicious apps are gathered as training set and with the help of Support vector machine algorithm and decision tree algorithm, comparison is made with training dataset and trained dataset. With this, we can predict the malware android apps nearly up to 93.2 % unknown or new malware mobile application. In this paper, SIGPID, Significant Permission Identification (SIGPID) system is implemented .SIGPID is used to enhance the apps permissions effectively and efficiently and also improves the accuracy in detecting malware applications. In this work, machine learning algorithms such as SVM and Decision Tree algorithms are utilized to make a comparison between training dataset and trained dataset .Support vector machine algorithms act as a classifier which is used to classify malicious application and benign app.This work is implemented with Confusion matrix with SVM, Confusion matrix with Decision Tree and Confusion matrix Naïve Bayes. It is observed that maximum malicious applications have been detected with Confusion matrix with Decision Tree.

Keywords - Android malicious applications, Malware detection, Machine learning, Mobile malware, Smartphone, SVM.

1. INTRODUCTION

The Android platform makes use of the permission system to limit applications privileges to secure the sensitive resources of the users. The developer is responsible for determining appropriately which permissions an application requires, but it is the responsibility of the user also to check properly before giving approval to the applications, as the applications requires user's approval of the requested permissions to access private or otherwise-restricted resources. According to recent studies, many users without understanding the permission preveliages simply grant them, thereby allowing an application to access sensitive/private information. Another laws that the user cannot decide to grant single permissions, while denying others. Many users, although an app might request a suspicious permission among much seemingly legitimate permission, will still accept for installation.

The Android security model [1] is based on permissions. An Android permission [2] provides restriction for confining access to a part of the code or to the data on the device. The limitation is imposed to protect critical data and code that could be misused to distort or damage user's experience. Permissions are also used to allow or restrict application access to restricted APIs and resources. For example, the Android 'INTERNET' permission is required by apps to perform network communications; so, opening a network connection is restricted by the 'INTERNET' permission. Furthermore, an application must have the 'READ CONTACTS' permission in order to read entries in a user's phonebook as well. To request for permission, the developer informs them using the Manifest file in declaring a "<uses-permission>" attribute. The "android:name" field specify the name of the permission in the code. Permission can be related with one of the following four protection levels:

1.1. Normal: A low-risk permission which allows applications to access API calls ('SET WALLPAPER') causing no harm to users.

1.2. Dangerous: A high-risk permission which allows applications to access potential harmful API calls('READ CONTACTS') such as leaking private user data or control over Smartphone device. Dangerous permissions are explicitly shown to the user before an app is installed and the user must decide to grant the permissions or not, determining whether the installation continues or fails, respectively.

1.3. Signature: A permission which is granted if its requesting application is signed with the same certificate as the application which clarify the permission is signed.

1.4.Signature-or-system: A permission which is granted only if its requesting application is in the same Android system image or is signed with the

same certificate as the application which clarify the permission is signed.

2. PROPOSED SYSTEM

In existing system, non machine learning methods were used to detect malicious applications based on properties, behavior and features, which were not efficient and reliable. It was difficult to identify newly created or updated malicious applications. Machine learning approaches are proposed and implemented in this work.

Significant Permission Identification (SIGPID) is implemented. This SIGID system enhances the accuracy and efficient detection of malware application. With the help of machine learning algorithms such as SVM and Decision Tree algorithms, which provides a comparison between training dataset and trained dataset. Support vector machine algorithms act as a classifier which is used to classify malicious application and benign app.

The proposed system enhances the detection of malicious applications as it uses machine learning algorithms [3] which very efficient, also able to detect newly created and updated malware android applications. Overall system architecture is shown in shown in figure 1, in the form of activity chart.

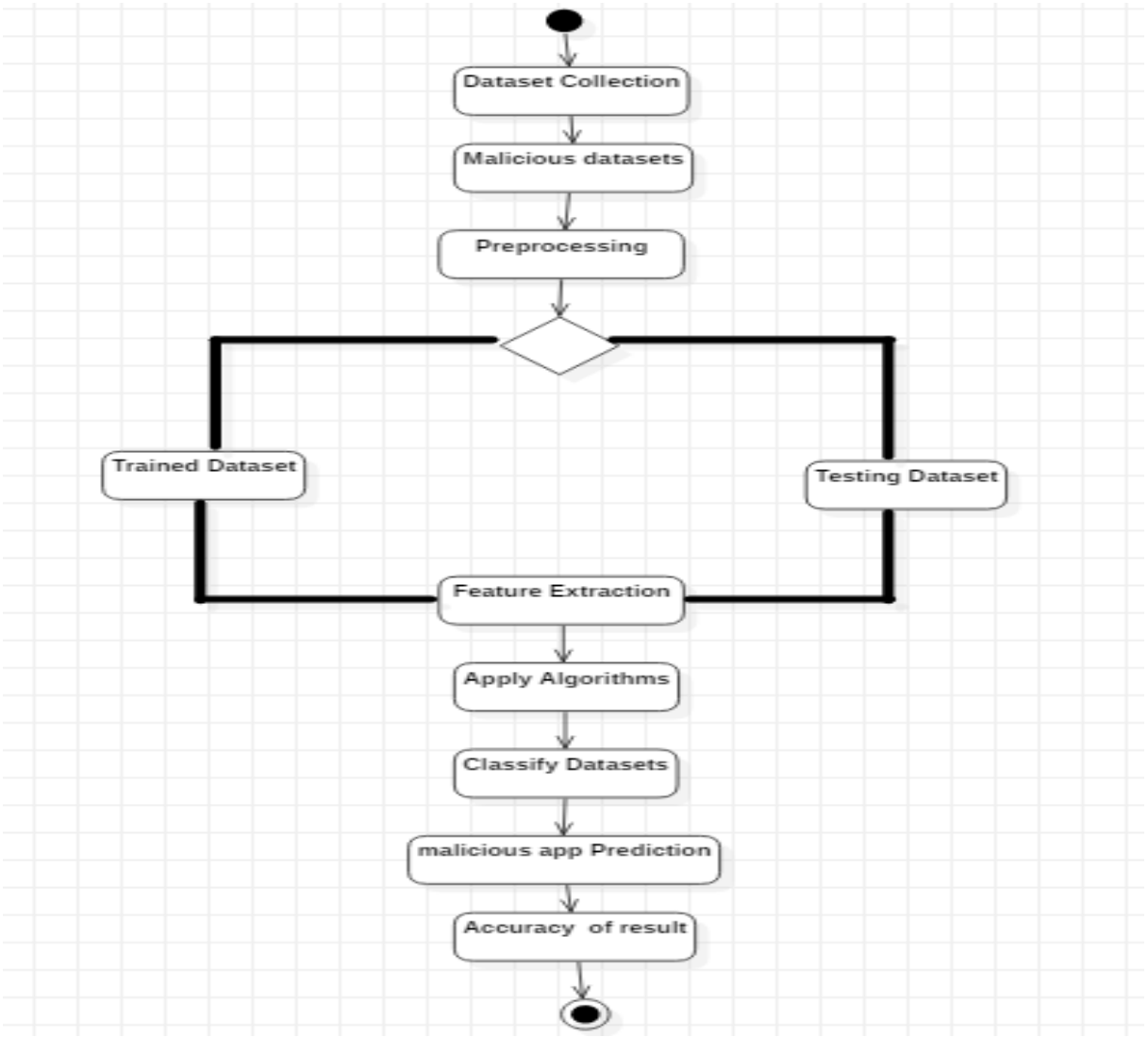

Figure 1. System Architecture

3. EXPERIMENTAL DESIGN

Our experimental design is done in two phases, i.e., Training Phase and Testing Phase shown in figure 2, which are further divided into different phases. These two phases, corresponding to the building of the decision tree and the development of the mobile application, represented in figure 2.

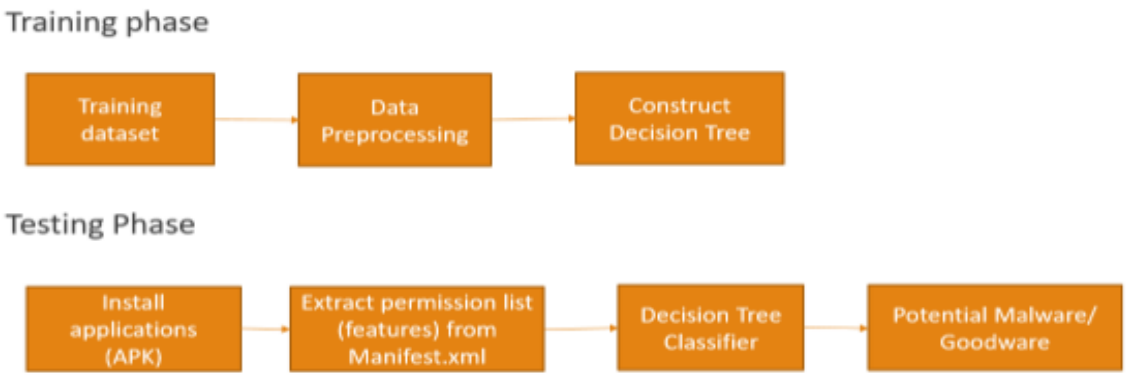

Figure 2. Experimental design

3.1. Phases for Data Preprocessing:

3.1.1. Formatting: The selected data may not be in a format that is compatible or suitable to work with. The data may be in a relational database, but flat file format is needed or the data may be in a proprietary file format, but relational database or a text file format is required.
3.1.2. Cleaning: Cleaning of data is removal or fixing of missing data. There may be incomplete data instances that do not have complete data for processing purpose. These types of incomplete data should be removed. And also, some of the attributes may contain sensitive information and such attributes require anonymization or deletion from the data entirely.
3.1.3. Sampling: If data is very lengthy, it takes more or longer time to run the algorithms and larger computational and memory. So small sample of the selected data can be taken and executed, which will be faster for exploring and prototyping solutions before running the whole dataset.

3.2. This work is implemented as four modules:

3.2.1. Permission
3.2.2. Combination of Permission
3.2. 3. Feature Extraction
3.2. 4. Classification

3.2.1. Permission

Permission module defines the existing Android malware from various aspects, that includes the permissions requested. The main aspect is the permissions that are extensively requested in both malicious and benign apps. Malicious apps request more frequently on the SMS-related permissions, such as 'READ SMS', 'WRITE SMS', 'RECEIVE SMS', and 'SEND SMS'. It is observed that malicious apps tend to request more permissions than benign apps. The android applications are classified into several categories such as social, entertainment, communication, tools and productivity, multimedia and video, puzzle and brain games. A method is provided that analyses plain and clear files in Android application by extracting four types of keyword lists:(1) Permission, (2) Intent filter (action), (3) Intent filter (category), and (4) Process name. This approach decides the apps as malicious or benign, based on the malignancy score by classifying individually permissions.

3.2.2. Combination of Permission

This provides a high-level contextual analysis and an exploration of Android applications based on the implementation of permission-based security models[4], [5]. And these models work by applying network visualization techniques and clustering algorithms. This method, when applied on networks helps to define irregular permission combinations requested by abnormal applications. Attributes such as sources, nature and implications of sensitive data are used on Android devices in enterprise settings. These attributes characterized [6] malicious apps and the risks they pose to enterprises. From the analysis of third-party applications [7], Permission additions dominate the evolution of third-party apps, of which Dangerous permissions tend to account for most of the changes. Malware detection can be done based on three metrics for evaluation: i) the occurrences of a specific subset of system calls, ii) weighted sum of a subset of permission that the application required, and iii) a set of combinations of permissions.

3.2. 3. Feature Extraction

A new method to detect malicious Android applications through machine learning techniques is by analyzing the extracted permissions from the application itself. The features that are used to classify are: i) the presence of tags uses-permission and uses-feature into the manifest and ii) the number of permissions of each application. These features are the permissions requested individually and the «uses-feature» tag. The possibility of detecting malicious Android applications [8] can be implemented based on permissions and 20 feature attributes from Android application packages.

3.2.4. Classification

In this method, by combining results from various classifiers, can be a quick filter to detect more suspicious applications. And a framework is proposed, that plans to develop a machine learning-based malware detection system on Android to detect malware applications [9] and to enhance security and privacy for Smart phone users. This system monitors various permission-based features and events obtained from the android applications, and analyses these features by using machine learning classifiers to classify whether the application is benign or malware. Once, the Support Vector Machine trained offline on a dedicated system and only it is transferred to the learned model to the Smartphone for detecting malicious applications.

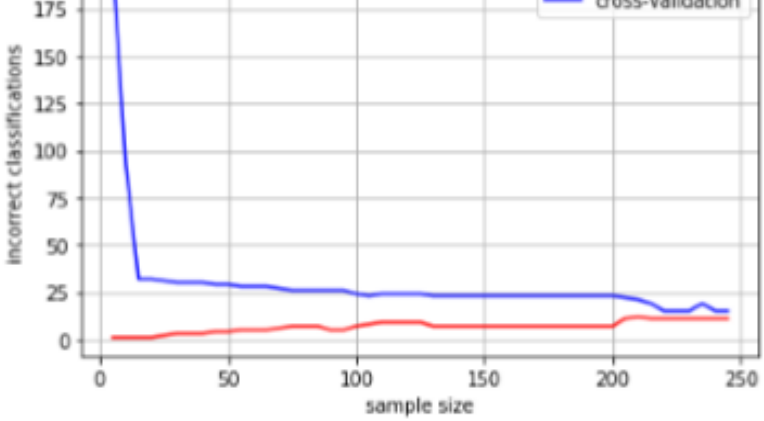

Figure 3. Learning Curve

The collected data is compared for true and false positive rate by the classifier [11], which is shown in figure 4. Figure 5 represents balanced Malware class. Figure 6 illustrates feature selection process. Malicious applications are detected based up on the parameters such as cache references, context switch, CPU usage, page faults, bus cycle branches etc., which are given in figure 6 and figure 7.

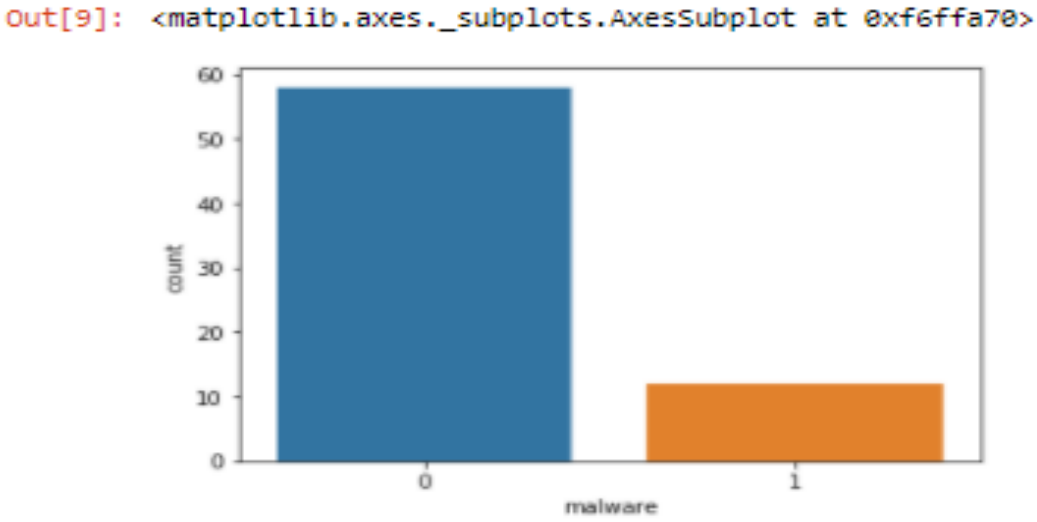

Figure 4. unbalanced class labels

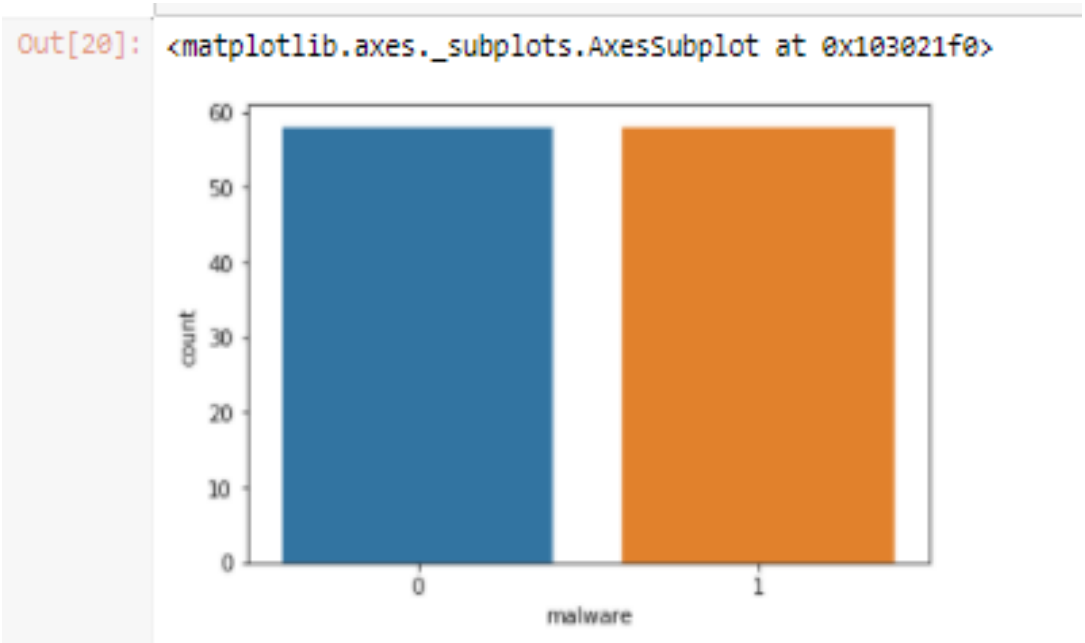

Figure 5. Visualization of Balanced Malware class

Out[44]:

	Specs	Score
9	branches	1.114483e+06
14	ref-cycles	2.882331e+05
6	stalled-cycles-backend-percent	1.255644e+05
11	bus-cycle	4.405044e+04
5	stalled-cycles-frontend-percent	4.003618e+04
13	cache-references	3.679843e+03
7	Instructions-per-cycle	2.790701e+03
12	cache-misses-percent	4.556774e+02
3	page-faults	1.169330e+02
10	branch-misses-percent	8.889021e+01

Figure 6. Feature Selection process

```
[0.06391337 0.04524854 0.01839569 0.06943939 0.02815262 0.01992447
 0.03237499 0.01862666 0.2106383  0.10583143 0.12077583 0.08900648
 0.08174854 0.04222666 0.05369705]
```

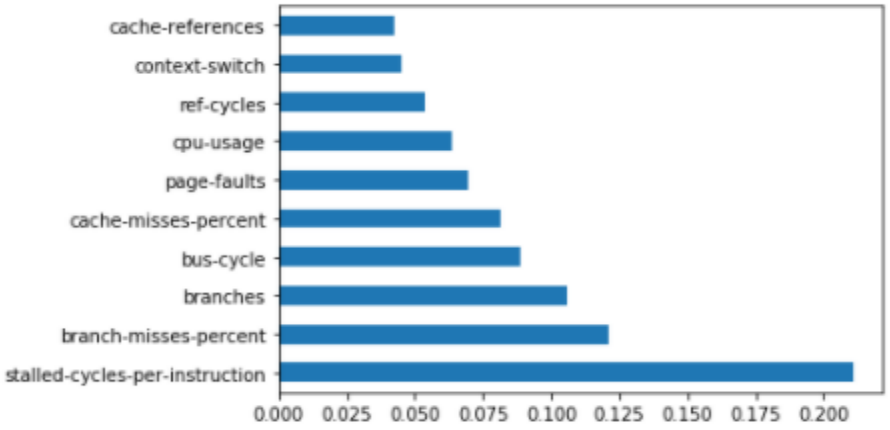

Figure 7. Parameters

Figure 8 shows the results with SVM and figure 9 represents results of Confusion Matrix with SVM in the form of graph.

```
In [54]: y_pred

Out[54]: array([1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1,
                0, 0], dtype=int64)

In [55]: model1=metrics.accuracy_score(y_test,y_pred)
         print(model1)

         0.875
```

Figure 8. Accuracy with SVM

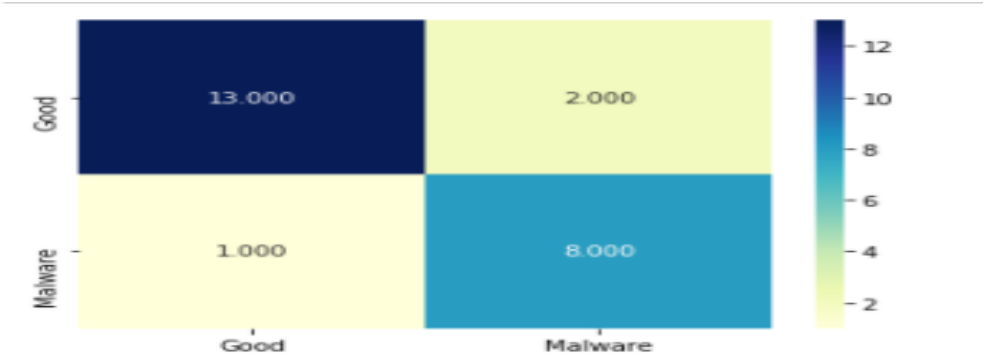

Figure 9. Confusion Matrix with SVM

Figure 10. represents Accuracy and Confusion matrix with Decision Tree. Figure 11 shows Accuracy and Confusion matrix with Naïve Bayes method and Figure 12 shows the comparative Results of SVM, decision tree classifier and Naïve Bayes method. From the comparison it is clear that with Confusion matrix with Decision Tree algorithm or method more malicious android applications [12] are detected.

```
In [61]: y_pred = tree.predict(X_test)
         y_pred

Out[61]: array([1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1,
                1, 0], dtype=int64)

In [62]: model2=metrics.accuracy_score(y_test,y_pred)
         print(model2)

         0.9583333333333334

In [63]: cnf_matrix = confusion_matrix(y_test,y_pred)

In [64]: labels = [0,1]
         sns.heatmap(cnf_matrix, annot=True, cmap="YlGnBu", fmt=".3f", xticklabels=labels, yticklabels=labels)
         plt.show()
```

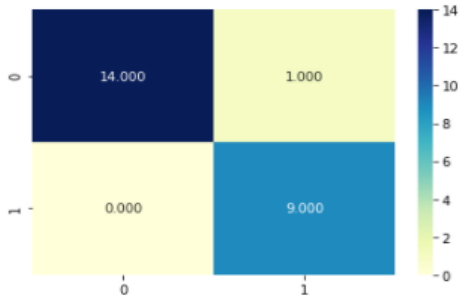

Figure 10. Accuracy and Confusion matrix with Decision Tree

```
In [68]: y_pred = clf.predict(X_test)
         y_pred

Out[68]: array([1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
                1, 0], dtype=int64)

In [69]: model3=metrics.accuracy_score(y_test,y_pred)
         print(model3)

         0.5416666666666666

In [70]: cnf_matrix = confusion_matrix(y_test,y_pred)

In [71]: labels = [0,1]
         sns.heatmap(cnf_matrix, annot=True, cmap="YlGnBu", fmt=".3f", xticklabels=labels, yticklabels=labels)
         plt.show()
```

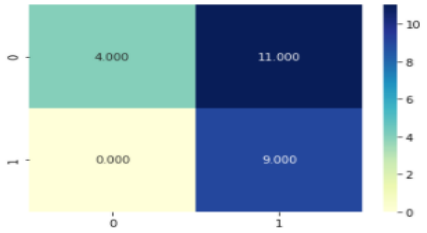

Figure 11. Accuracy and Confusion matrix with Naïve Bayes

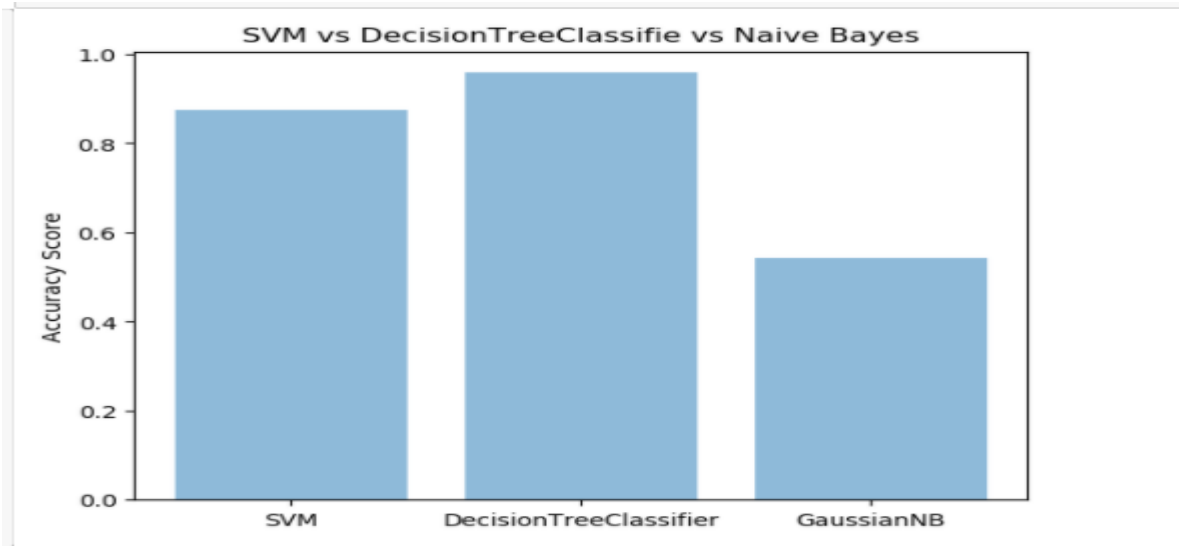

Figure 12. Comparison Results

CONCLUSION AND FUTURE WORK

In this work, Significant Permission Identification (SIGPID) system is implemented which scans applications on the phone at any time, and alert the user when an installation or app update occurs. This is an important step in preventing Android malware, because this application warns or alerts the user about all the possible dangerous applications, allowing them to scrutinize the applications carefully that they trust more. This work is implemented in Python, which gives good accuracy. And can also be implemented in Java, which may enhance the accuracy of classifier.

REFERENCES

[1]. W. Enck, M. Ongtang, P. McDaniel, "Understanding Android Security".

[2]. B. P. Sarma, N. Li, C. Gates, R. Potharaju, C. Nita-Rotaru, and I. Molloy, “Android permissions: a perspective combining risks and benefits,” 2012.

[3]. Urcuqui, C., and A. Cadavid. "Machine learning classifiers for Android malware analysis." Proceedings of the IEEE Colombian Conference on Communications and Computing. 2016.

[4]. C.-Y. Huang, Y.-T. Tsai, and C.-H. Hsu, “Performance Evaluation on Permission-Based Detection for Android Malware,” 2013.

[5]. Franklin Tchakount´, Computers & Security “Permission-based Malware Detection Mechanisms on Android: Analysis and Perspectives”, 2014.

[6]Y. Zhou and X. Jiang, “Dissecting android malware: Characterization and evolution,2012.

[7]. A. P. Felt, K. Greenwood, and D. Wagner, “The effectiveness of install-time permission systems for third-party applications”,2010.

[8]. Franklin Tchakount´, Computers & Security “Permission-based Malware Detection Mechanisms on Android: Analysis and Perspectives”, 2014.

[9]. Aung, Zarni, and Win Zaw. "Permission-based android malware detection."International Journal of Scientific and Technology Research2.3 (2013): 228-234.

[10]. Urcuqui, C., and A. Cadavid. "Machine learning classifiers for Android malware analysis." Proceedings of the IEEE Colombian Conference on Communications and Computing. 2016.

[11]. G. Canfora, F. Mercaldo, and C. A. Visaggio, “A classifier of malicious android applications,” 2013.

[12]. Liu, K.; Xu, S.; Xu, G.; Zhang, M.; Sun, D.; Liu, H. A Review of Android Malware Detection Approaches Based on Machine Learning. IEEE Access 2020, 8, 124579–124607.