Detecting Malicious Apps in Android Devices using SVM, Random Forest & Decision Trees

E. Sanjana, M. Srikanth Sagar, Deekshitha Nalla, Bonthu Meghana, Anusri Katragadda

Abstract*: *In recent years, the usages of smart phones are increasing steadily and also growth of Android application users are increasing. Due to growth of Android application user, some intruder are creating malicious android application as tool to steal the sensitive data. We need an effective and efficient malicious applications detection tool to handle new complex malicious apps created by intruder or hackers. This project deals with idea of using machine learning approaches for detecting the malicious android application. First we have to gather dataset of past malicious apps as training set and with the help of Support vector machine algorithm and decision tree algorithm make up comparison with training dataset and trained dataset we can predict the malware android apps upto 93.2 % unknown / New malware mobile application. By implementing SIGPID, Significant Permission Identification (SIGPID).The goal of the sigid is to improve the apps permissions effectively and efficiently. This SIGPID system improves the accuracy and efficient detection of malware application. With the help of machine learning algorithms such as SVM, Random Forest Classifier and Decision Tree algorithms we make a comparison between training dataset and trained dataset to classify malicious application and benign app.

Keywords : SVM(Supprt Vector Macchine), Random Forest Classifier, (SIGPID) Significant Permission Identification, Decision Tree algorithm.

I. INTRODUCTION

As we know machine Learning is one of the intelligent methodologies that have shown promising results in the domains of classification and prediction. This project deals with idea of using machine learning approaches for detecting the malicious android application. First we have to gather the dataset of past malicious apps as training set and with the help of Support vector machine algorithm, Random Forest classifier and decision tree algorithm[2] make up comparison with training dataset and trained dataset we can predict the malware android apps upto 93.2 % unknown / New malware mobile application.

Revised Manuscript Received on May 06, 2020.

E. Sanjana , Student, Department of Computer Science and Engineering, Mahatma Gandhi Institute of Technology, Hyderabad, India sanjsanjana7@gmail.com

M. Srikanth Sagar, Assistant professor, Department of Computer Science And Engineering Mahatma Gandhi Institute of Technology, Hyderabad, India msrikanthsagar_cse@mgit.ac.in

Deekshitha Nalla, Student, Department of Computer Science and Engineering, Mahatma Gandhi Institute of Technology, Hyderabad, India ndeekshitha26@gmail.com

Bonthu Meghana, Student, Department of Computer Science and Engineering, Mahatma Gandhi Institute of Technology, Hyderabad, India 17meghanabonthu@gmail.com

Anusri Katragadda, Student, Department of Computer Science and Engineering, Mahatma Gandhi Institute of Technology, Hyderabad, India anusrikatragadda99@gmail.com

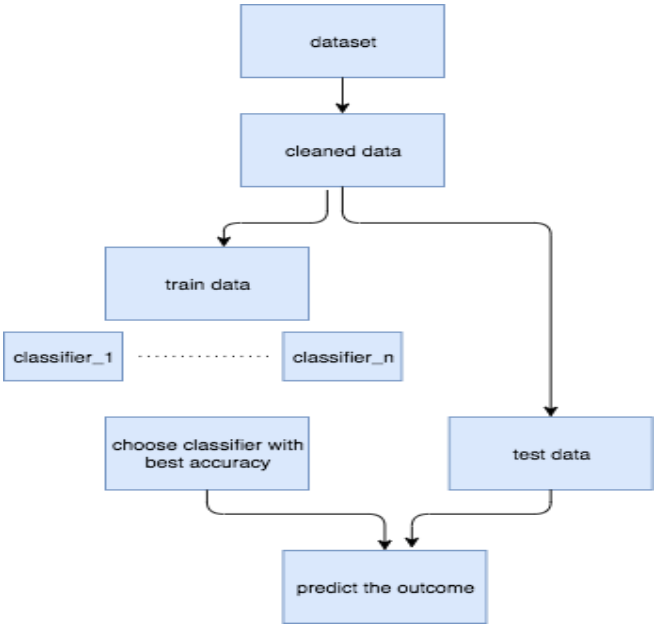

Fig. 1.Proposed system for Malicious Application Prediction

In proposed system, we implement SIGPID, Significant Permission Identification .The goal of the sigpid is to improve the apps permissions effectively and efficiently[8]. The shows the Proposed system for malicious application prediction. This SIGPID system improves the accuracy and efficient detection of malware application. We will use different classifiers for results prediction, compare the result and choose the optimal classifier for proper app prediction. Classification Models like Decision Trees, Random Forest and SVM will be used for primary classification of the data. We will then apply the model which gives us the best accuracy.

II. METHODOLOGY

A. Architecture

Here, we explain our approach to the problem in detail, including the definitions and the mechanism of various algorithms used to model the teams in order to predict the application and to find out the best algorithm. We use three supervised algorithms in order to predict the results of application.

The overall system is designed in the following phases:

1) Dataset Collection
2) Training the model
3) Predictions
4) Evaluation of the model
5) Choosing the best model to predict the app

1) Dataset Collection

I am using Android Malicious datasets for this project. The dataset is from the data science community Kaggle.This dataset is used to perform exploratory data analysis.

2) Training the model

Various models such as decision trees, Random Forest, SVM are trained using the data and the predictions are made.

3) Predictions

Based on the features the models predict the outcomes on the test data which accounts for the 20% of the data.

4) Evaluation of the model

The test data is used to compare the predictions and the actual outcomes of the app and we can calculate the accuracy by measuring the root mean square error values.

5) Choosing the best model to predict the winner of the tournament

The model with the best accuracy is chosen and the model is applied to predict the malicious app.

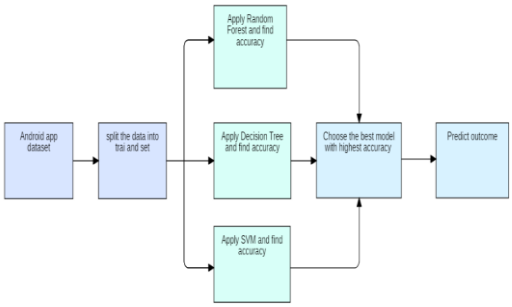

Fig. 2.Architecture of the prediction model

The above Fig. 2 shows the architecture of the prediction model.

B. Decision Tree

A decision tree uses a tree-like chart or model of decision and their conceivable results, including possible outcomes, asset expenses, and utility[2]. Decision tree is a of supervised learning calculation it has a pre-characterized target factor that is mostly used in classifications.

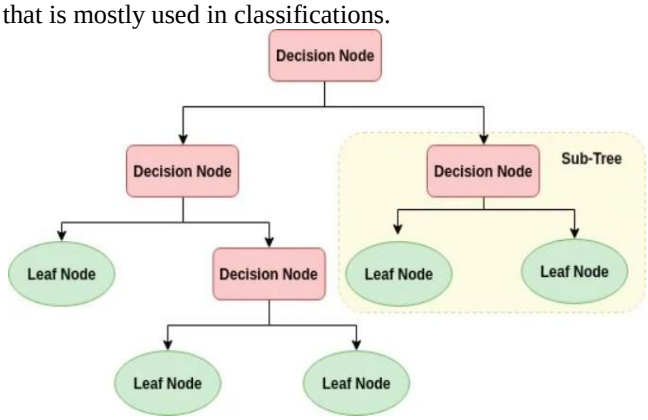

Fig. 3.Functioning of Decision trees

It works for both data divided in categories and continuous input and output factors. Here, we split the or test into at least two similar sets based on most differentiated input factor. This is shown in the above Fig. 3.

C. Random Forest Classifier

Random forest is a classifier that comprises of a large number of decision trees which act as an ensemble[2]. Every decision tree in a Random Forest is given a class. prediction. The class which gets the most votes is our class prediction. The basic concept behind Random Forest is a powerful one. Random Forest works better than other models because many uncorrelated models i.e. trees operate as a committee whose performance is better than individual models.

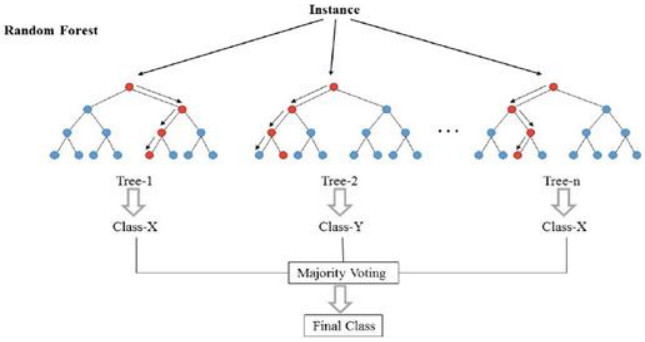

Fig. 4.Functioning of Random Forest Classifier

The above Fig. 4 shows the functioning of Random Forest Classifier. The key to the Random Forest classifier is the low correlation between the models present in it. The models with low correlation produce more accuracy than the individual accuracy prediction. This is the case because trees tend to protect each other from their own errors. Some trees might be wrong while others right. Therefore as group trees will move in the right direction.

D. Support Vector Machine

The main aim of the support vector machine algorithm is to observe and find a hyperplane in an N-dimensional space that distinctly classifies all the data points[3]. There are many possible hyperplanes to separate the data points into two classes.

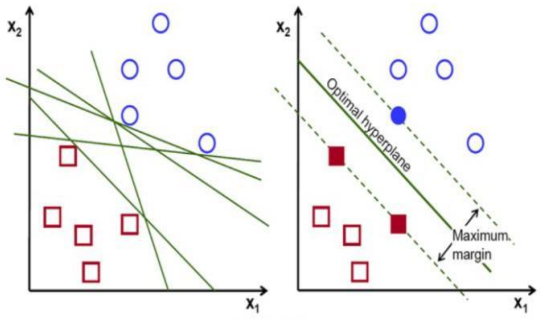

Fig. 5.Possible Hyperplanes

There are many possible hyperplanes to separate the data points into two classes. This is shown in the above Fig. 5.The hyperplane which has the greatest distance between data points of both the classes is chosen. The margin distance between the data points should be maximized so that it provides some reinforcement. So, later the data points can be classified with more confidence.

III. RESULTS

After the data has been pre-processed and analysed, it is split into training and test data in the ratio of 70% to 30%. The training data is used by the machine learning model to learn and test data is used to make

predictions and to evaluate the efficiency of the model. Various ML models such as SVM, Decision tree and Random Forest are applied and the model with the best accuracy is chosen for further predictions.

A. Evaluation of Random Forest Classifier

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,Y, test_size = 0.2, random_state=0)
```

```
rf.fit(X_train,y_train)
```

```
ypred=rf.predict(X_test)
```

```
model3=metrics.accuracy_score(y_test,ypred)
print(model3)
```

```
0.9375
```

Fig. 6. Accuracy of Random Forest Classifier

The accuracy obtained by applying this model has given us an accuracy of 93% for produced prediction. This is shown in the above Fig. 6.

Confusion Matrix for Random Forest Classifier

```
cnf_matrix = confusion_matrix(y_test,ypred)
```

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

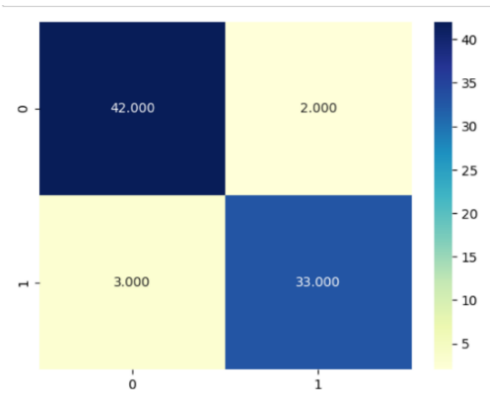

Fig. 7.Confusion matrix of Random Forest classifier

The above Fig. 7 shows the confusion matrix of Random Forest Classifier.

B. Evaluation of Support Vector Machine

```
from sklearn import svm
support = svm.LinearSVC(random_state=20)
```

```
support.fit(X_train,y_train)
```

```
y_pred = support.predict(X_test)
```

```
model1=metrics.accuracy_score(y_test,y_pred)
print(model1)
```

```
0.95
```

Fig. 8.Accuracy of Support Vector Machine

The accuracy obtained by applying this model has given us an accuracy of 95% for produced prediction. This is shown in the above Fig. 8.

Confusion Matrix for Support Vector Machine

```
cnf_matrix = confusion_matrix(y_test,y_pred)
```

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

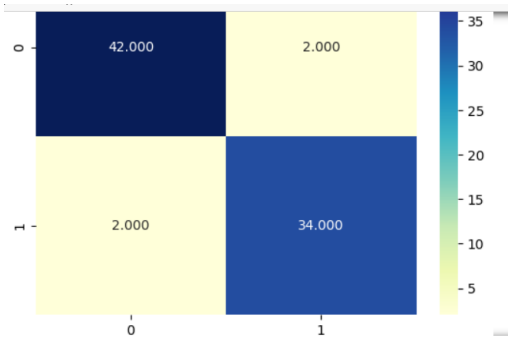

Fig. 9.Confusion matrix of Support vector machine

The above Fig. 9 shows the confusion matrix of Support vector machine.

C. Evaluation of Decision Tree Classifier

```
tree = DecisionTreeClassifier()
```

```
tree.fit(X_train,y_train)
```

```
y_pred = tree.predict(X_test)
```

```
model2=metrics.accuracy_score(y_test,y_pred)
print(model2)
```

```
0.925
```

Fig. 10.Accuracy of Decision Tree Classifier

The accuracy obtained by applying this model has given us an accuracy of 92.5% for produced prediction. This is shown in the above Fig. 10.

Confusion Matrix for Decision Tree Classifier

```
cnf_matrix = confusion_matrix(y_test,y_pred)
```

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

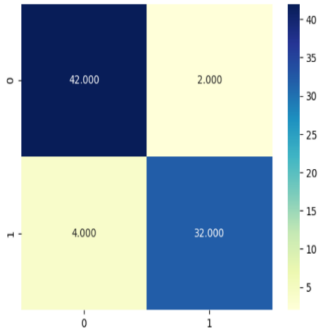

Fig. 11.Confusion matrix of Decision Tree Classifier

The above Fig. 11 shows the confusion matrix of Decision Tree Classifier.

IV. CONCLUSION AND FUTURE SCOPE

As there are many ways and techniques in which we can predict the app outcomes. We have performed data mining steps such as exploratory analysis and feature extraction to select the most relevant feature for prediction, data manipulation and then choose a Machine learning model and deploying it on the dataset. For the given set of attributes Support Vector Machine performed the best compared[8] to other classification algorithms like Decision Trees and Random Forest.

Future work for the project will include an increase in the accuracy of the classifier, migrating the Python portions of this project to Java, and integrating more advanced methods of detecting malicious behavior such as looking at API calls[4]. Benefit of the decision tree classifier is its speed. It can serve as a preliminary screen for more advanced but slower methods, to focus the applications they will inspect. Lastly, taking into account application categories such as being a game or email-client would also help detect suspicious permissions and behavior.

REFERENCES

1. J. Song, C. Han, K. Wang, J. Zhao, R. Ranjan, L. Wang, "An integrated static detection and analysis framework for Android", Pervas. Mobile Comput., vol. 32, pp. 15-25, Oct. 2016.
2. W. Wang, X. Wang, D. Feng, J. Liu, Z. Han, X. Zhang, "Exploring permission- induced risk in Android applications for malicious application detection", IEEETrans. Inf. Forensics Security, vol. 9, pp. 1869-1882, Nov. 2014.
3. G. Dai, J. Ge, M. Cai, D. Xu, W. Li, "SVM-based malware detection for Android applications ", Proc. 8th ACM Conf. Secur. Privacy Wireless Mobile Netw., Jun. 2015.
4. S. Sheen, R. Anitha, V. Natarajan, "Android based malware detection using a multifeature collaborative decision fusion approach", Neurocomputing, vol. 151, pp. 905-912, Mar. 2015.
5. N. Peiravian and X. Zhu, “Machine learning for android malware detection using permission and api calls,” in ICTAI, 2013, pp. 300–305.
6. B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero, P. G. Bringas, and G. Álvarez, ‘‘PUMA: Permission usage to detect malware in Android,’’ in Proc. Int. Joint Conf. CISIS-ICEUTE-SOCO Special Sessions, 2013, pp. 289–298.
7. Machine Learning-Based MaliciousApplication Detection of Android LINFENG WEI 1, WEIQI LUO1, JIAN WENG1, YANJUN ZHONG1.
8. Significant permission identification for android alware detection by Lichao Sun, Zhiqiang Li, Yu Pan.

AUTHORS PROFILE

Ms. E. Sanjana, a Final year Student of Bachelors in Engineering in the field of Computer Science at Mahatma Gandhi Institute of Technology, Hyderabad. She has worked as an intern for several companies and has developed projects in IOT, Web technologies, Machine learning etc. Her areas of interests in research include Data Mining, Data Analytics, Web Technologies and IOT, Machine learning.

Mr. M Srikanth Sagar , an Assistant Professor in the Department of Computer Science and Engineering at Mahatma Gandhi Institute of Technology, Hyderabad. He has a work experience of over 7.5 years. He has worked with TCS and Cognizant software companies and he is in academics. His areas of interests in research include Data Sciences, Machine learning.

Ms. Bonthu Meghana, a final year student of Bachelors of Technology in the field of Computer Science at Mahatma Gandhi Institute of Technology, Hyderabad. She has developed projects in the fields of Android Application Development, IOT, machine learning . Her areas of interests in research include Python, Data mining.

Ms. Anusri Katragadda, a Final year Student of Bachelors in Engineering in the field of Computer Science at Mahatma Gandhi Institute of Technology, Hyderabad. She has developed projects in IOT, Machine learning etc. Her areas of interests in research include Data Mining, Data Analytics, Web Technologies and IOT,Machine learning.

Ms. Deekshitha Nalla, a Final year Student of Bachelors in Engineering in the field of Computer Science at Mahatma Gandhi Institute of Technology, Hyderabad. She has developed projects in Web technologies,Machine learning etc. Her areas of interests in research include Data Mining, Data Analytics, Web Technologies and Machine learning