University of New Haven

Tagliatela college of Engineering

Master's in Data Science

Machine Learning and Data Analysis I

Android Malware Recognition system:

Mobile Malware recognition system using Ensemble Learning.

Name : Raja Muppalla

Under the guidance of,

Prof : Travis Millburn

tmillburn@newhaven.edu

TABLE OF CONTENTS

TABLE OF FIGURES

1. Introduction

In this digital era, it is the need of the hour for everyone to have their smartphones, tablets, and other multimedia devices. The devices are used for browsing the internet, sending text messages, making calls, and numerous other things. These things are done on the devices with the help of applications. For an android device, we have a play store that has billions of applications to be downloaded and installed on our handheld device. With the increased number of users on the application front, there is a risk of illegal intrusion. With the evolution of technology, hackers have also evolved from their manual hacking methods to the bot which are controlled by the network to extract data from the device and steal from the device holder. Malware is sophisticated and challenging to combat. The users will face the network-controlled ransom ware, wiper malware, and the adversaries that are used to break down the security measures.

Ensemble learning has come by expended recognition in the machine learning community and come out as a well-liked method of AI being applied in many fields. DL classifiers have motivated remarkable practical perspectives in image classification, natural language processing, etc.

In this paper, we are focussed on taking benefit of the advantages of the ensemble model by enhancing a traditional and efficient classification model: Decision Trees with gradient descent. Our process will also focus on feature extraction and feature selection; the model is then trained and tested. Its performance is evaluated using metrics like accuracy and F1 score. The model's performance is then compared with traditional models like SVM and Naïve to see if the ensemble models outperform traditional models.

1.1 Background

Shabtai et al. proposed Andromaly, a host-based Android malware recognition solution that employs dynamic analysis. Anomaly continuously monitors various features and events like CPU consumption, several packets sent several running processes, keyboard/touch-screen pressing, etc. Machine learning anomaly detectors are then applied to classify the collected data into normal or abnormal. AntiMalDroid was proposed by Zhao et al. AntiMalDroid is a dynamic

analysis behaviour-based malware recognition framework that uses logged behaviour sequence as features for SVM model training and recognition.

[Arp et al. 2014] proposed (DERBIN) a lightweight static analysis framework that extracts a set of features from the app's AndroidManifest.xml (hardware components, requested permissions, App components, and filtered intents) and disassembled code (restricted API calls, user agreements, blocked API calls, network addresses) to generate a joint vector space. Support Vector Machines (SVM) was applied to the dataset to learn a separation between benign and malicious apps.

[Chan and Song 2014] selected permissions and APIs features with Information Gain to train different classifiers: Naive Bayes, SVM with SMO algorithm, RBF Network, Multi-Layer Perceptron, Liblinear, decision tree, and Random Forests.

1.2 Research Question

- How accurately our proposed work can identify the new mobile application malware?
- How the feature selection techniques help us to achieve better accuracy?

2. Literature Review

With the advent of Android phones, the use of mobile phones has increased rapidly. Together with audio and video calls over the Android, people have used several mobile applications downloaded from the Google Play stores as well as the other application agents. It is also a common trend that people pair their Android phones with different web applications or with other individuals in public places that will increase the probability of data-stealing through interconnected mobile malware applications. These malware applications are designed by creative hackers that require intelligent processing and recognition agents to detect. In the last many decades, traditional algorithms were used for this purpose that fails to counter the malware problems because of their computational inadequacy and featureless solution. But from the last several years, Machine learning has produced critical results with their feature selection, and malware classification approaches. This work will review the essential classification methods

that were used to detect the anomalies in the mobile application domain together with the brief background of some traditional approaches.

The goal of this review is to emphasize how tree-based classifiers' or ensemble classifiers are best for classifying the anomalies and malware in mobile phones and its usability.

3. Research Method and Specification

For this paper, we are using Malgenome mobile malware dataset provided by NorthCarolina State University. The dataset consists of 215 mobile application features with the 3800 samples. The dataset features involve trsact, onService-Connected, Service Connection and send SMS etc. which will be used for our analysis.

The first step to begin with a machine learning model is to preprocess the data. Then we perform feature extraction where we extract the features by checking their significance in for the model. We will use the Chi-square test to extract essential features from the dataset. This is required as all the features in the dataset may not be equally important. Once all these steps are performed, then the data is separated into training and test subsets of the data. These are nothing but randomly selected observations from the data. Then we will be training machine learning model. For this paper, we will use ensemble learning methods, whereby we combine two different machine learning models to achieve accurate and better results. For this paper, we are combining the Decision tree with Gradient boosting algorithm, where the results will be compared with other algorithms which include Support vector machine and naive Bayes algorithm. Accuracy andF1-score and training time of a model will be used as metrics.

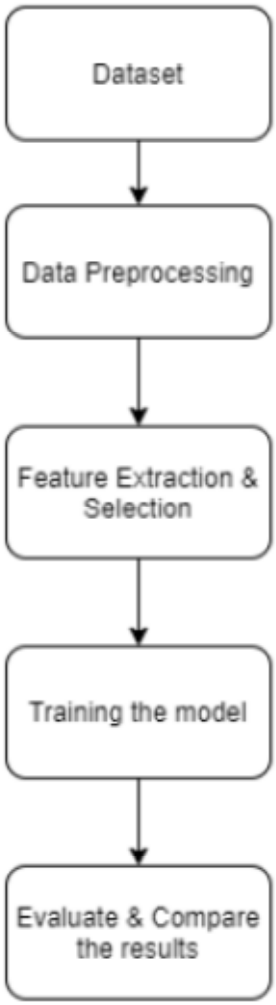

Figure : Proposed Model

3.1 Gradient boosting algorithm

Gradient boosting is a machine learning technique for regression and classification problems that build a model attendance model in a synchronized form with weak predictive models, with simple decision perspectives. (Wikipedia definition)

The goal of any supervised learning algorithm is to define the risk and minimize it. Let us see how the gradient boosting algorithm calculates. We want our loss function (MSE) to be as low as possible. By using Gradient Descent and updating our future based on the study rate, we can find the lowest values of MSE. The logic behind increasing the gradient is very simple (can be understood consciously without using mathematical notation). The basic premise of linear regression is that the sum of its residues is 0, which means that the debris must spread randomly around zero. Therefore, it is advisable to repeatedly use the patterns in the understanding of

debris behind the gradient-increasing algorithm and reinforce a pattern with weak predictions. Once we have reached a point where there is no design to model the waste, we can stop modelling the waste (otherwise it may lead to over-fitting). According to the algorithm, we reduce our loss when the test loss reaches its minimum.

Steps for Gradient Boost Algorithm

First, we first model the data using simple models and analyze the data for errors.

These errors indicate data points that fit the general model.

For the latter models, we pay special attention to those who find it difficult to get the data right.

In the end, we unite all the predictors, giving each predictor some weight.

3.2 Decision Tree Algorithm

The Decision Tree is a supervised learning technique that is useful for classification and regression problems, but it is preferred to solve taxonomy problems. It is a tree-structured taxonomy, where the internal nodes represent the characteristics of the dataset, the branches represent the decision rules, and each leaf node represents the result. From the decision point of view, there are two nodes, namely Decision Node and Leaf Node. Decision nodes are used for decision-making and multiple branches, whereas leaf nodes do not have the output output and more branches of those decisions.

Decisions are made or tested based on the characteristics of a given dataset. It is a graphical representation of all possible solutions to a problem / decision based on the given circumstances. This is called a decision view because, like the tree, it starts from the root node and expands into more branches, forming a tree-like structure. To construct the tree, we use the CART algorithm, which represents the taxonomy and regression tree algorithm. The decisive perspective asks a question, based on the answer (yes / no), which divides the tree into subtypes.

3.3 Support Vector Machine

Support vector machine or SVM is the most popular supervised learning algorithm used for classification and regression problems. However, it is mainly used for classification problems in machine learning. The goal of the SVM algorithm is to create the best line or decision boundary that can be divided into ecological space classes so that new data points can be easily included in the appropriate category in the future. This best decision is called a hyperplane. SVM selects the extreme points / vectors that help to create the hyperplane. These extreme cases are called support vectors; hence the algorithm is called the support vector machine.

There are two types of SVMs:

Linear SVM: Linear SVM is used for linearly separable data, i.e. if the dataset can be divided into two classes with a single straight line, such data is called linearly separable data and the classification is called linear SVM classification. We have used Linear SVM for our classification.

Non-Linear SVM: Non-linear SVM is used for non-linear separated data, i.e. if the dataset cannot be sorted by a straight line then such data is called non-linear data and the classification is called non-Linear SVM classification.

3.4 Naïve Bayes Classification

Naive base algorithm is a learning algorithm that is monitored based on Bayes theory and used to solve classification problems. It is mainly used in text classification, which contains a large number of training datasets. Now bias classification is a simple and efficient classification algorithm that helps in creating fast machine learning models that can make quick predictions. This is the probability classification, which is the probability of an object.

Naive: It is called naive because it thinks that the occurrence of a particular symptom is different from other symptoms. When the fruit is identified based on colour, shape and taste, the red, spherical and sweet fruits are identified as apples. So, each feature contributes individually to identify it as an interdependent apple.

Bayes: It is called Bayes because it is based on the principle of Bayes theory.

4. Result Analysis

The comparison of all the four algorithms is made between Decision Tree, Gradient Boost, Support Vector Machine and Naïve Bayes for the evaluation metrics of accuracy, f1-Score and training time. The steps for doing the comparison and fining out the malicious data in android system is as follows:

1. Python libraries are imported to train the model with the entire four algorithms.
2. The .CSV format is loaded for model training purposes.
3. Shape of the dataset is defined, which has 3799 rows and 216 columns.
4. Type is the label that represents if an application is a malware or not, as we can see this dataset is balanced.
5. Benign and malicious malware activities are observed.
6. The plot depicts the various permission from various application in the android system.
7. The dataset is segregated into training and testing subset before training the models with all the algorithms.
8. Model training and evaluation happens.

```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report
from sklearn.ensemble import BaggingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import cohen_kappa_score


from sklearn import preprocessing
#import torch
from sklearn import svm
from sklearn import tree
import pandas as pd
import pickle
import numpy as np
import seaborn as sns
```

Figure: Importing python Libraries

```
import pandas as pd
df = pd.read_csv('malgenome215dataset1260malware2539benign.csv')
df
```

	transact	bindService	onServiceConnected	ServiceConnection	android.os.Binder	READ_SMS	attachInterface	WRITE_SMS	TelephonyManager.getSubscril
0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	1	0	0	
2	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	1	0	1	
4	1	1	1	1	1	1	1	1	
...	...	...	...	...	...	...	...	...	
3794	0	1	1	1	0	0	0	0	
3795	1	1	1	1	1	0	1	0	
3796	0	0	0	0	0	0	0	0	
3797	1	1	1	1	1	0	1	0	
3798	0	0	0	0	0	0	0	0	

3799 rows × 216 columns

Figure: Loading the dataset

```
#Checking the shape of the dataset which has 3799 rows and 216 columns
df.shape
```

```
(3799, 216)
```

Figure: shape of the dataset

```
#number of benign and malicious
#Type is the label that represents if an application is a malware or not, as we can see this dataset is balanced.
df.type.value_counts()
```

```
1    2539
0    1260
Name: type, dtype: int64
```

Figure: Type labelling

```
pd.Series.sort_values(df[df.type==1].sum(axis=0), ascending=False)[1:11]
```

```
Binder                          2198
IBinder                         2174
android.os.IBinder              2174
INTERNET                        2098
Ljava.lang.Object.getClass      2065
android.content.pm.PackageInfo  2008
ACCESS_NETWORK_STATE            1943
onBind                          1813
Ljava.lang.Class.forName        1788
android.os.Binder               1781
dtype: int64
```

Figure: Malicious activities from the dataset

```
pd.Series.sort_values(df[df.type==0].sum(axis=0), ascending=False)[:10]
```

```
INTERNET                            1232
Binder                              1189
android.os.IBinder                  1189
IBinder                             1189
READ_PHONE_STATE                    1179
onBind                              1170
TelephonyManager.getDeviceId        1095
android.intent.action.BOOT_COMPLETED  1061
android.content.pm.PackageInfo      1052
Ljava.lang.Object.getClass          1042
dtype: int64
```

Figure: Benign activities from the dataset

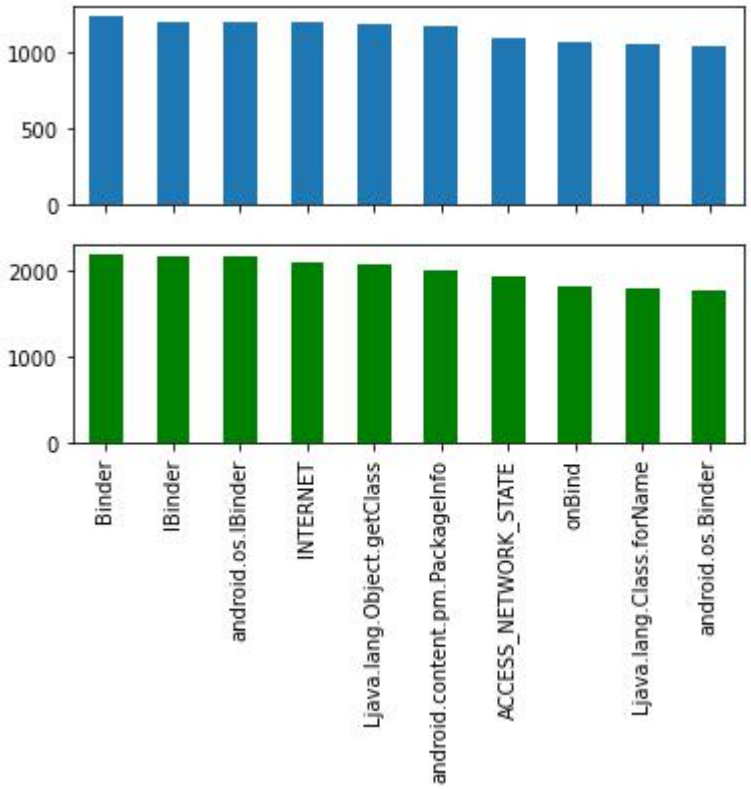

Figure: The plot depicts the various permission from various application in the android system

The above figure depicts the plot for the permission of various applications in the android system. We see Binder has highest number of permissions and rest all the application tends to have similar permission index.

```
naive_bayes
0.675
              precision    recall  f1-score   support

           0       0.98      0.50      0.66       486
           1       0.53      0.98      0.69       274

    accuracy                           0.68       760
   macro avg       0.75      0.74      0.67       760
weighted avg       0.82      0.68      0.67       760

Training time: 6.204121351242065s
```

Figure: Naïve Bayes Classifier

```
DecisionTreeClassifier()
0.9736842105263158
              precision    recall  f1-score   support

           0       0.95      0.97      0.96       245
           1       0.98      0.98      0.98       515

    accuracy                           0.97       760
   macro avg       0.97      0.97      0.97       760
weighted avg       0.97      0.97      0.97       760

Training time: 5.383313417434692s
```

Figure: Decision Tree Classifier

```
SVC(kernel='linear')
Accuracy: 0.9881578947368421
              precision    recall  f1-score   support

           0       0.98      0.50      0.66       486
           1       0.53      0.98      0.69       274

    accuracy                           0.68       760
   macro avg       0.75      0.74      0.67       760
weighted avg       0.82      0.68      0.67       760

Training time: 6.319193124771118s
```

Figure: SVC Classifier

```
GradientBoostingClassifier(learning_rate=0.5, max_depth=2, max_features=2,
                           n_estimators=20, random_state=0)
Classification Report
              precision    recall  f1-score   support

           0       0.97      0.86      0.91       249
           1       0.94      0.99      0.96       511

    accuracy                           0.95       760
   macro avg       0.95      0.93      0.94       760
weighted avg       0.95      0.95      0.95       760
```

Figure: Gradient Boosting Algorithm

The result from all the algorithms are compared and values of accuracy, f1-score and training timing is observed.

Algorithm	Accuracy	f1-Score	Training Time
Decision Tree	0.97	0.98	5.383
Gradient Boost	0.95	0.96	6.939
Support Vector Machine	0.98	0.69	7.249
Naive Bayes	0.67	0.69	6.204

Figure: Comparison table for the algorithms

The above table indicates that the support vector machine has the highest accuracy but has maximum duration of training time. The accuracy of naïve Bayes classifier is least. The decision tree classifier takes least amount of training time as compared to other three algorithm models.

5. Conclusion and Future Work

The conclusion of the project can be summarised as the malicious and Benign malwares are present in any android system. These malwares can be analysed with the help of static and dynamic methods. The static method just involves loading the data and analysing it but the dynamic method involves complex steps of web scrapping the url data. In our project we concentrated on static methods and observed the use of various machine learning algorithms like decision tree, gradient boost, support vector machine and naïve bayes classifiers.

The future work involves the use of dynamic method for malware classification of the android system. Other machine learning algorithms and techniques can also be used like neural networks, time series algorithms.

References

Sumit Mishra,2020 Soumyadeep Thakur, Manosij Ghosh, Sanjoy Kumar Saha, (2012), An Autoencoder Based Model for Detecting Fraudulent Credit Card Transaction

Fairuz Amalina Narudin Fairuz Amalina Narudin et al. Evaluation of machine learning classifiers for mobile malware recognition". In: Soft Computing 20.1 (2016), pp. 343{357.

Tanmoy Chakraborty Tanmoy Chakraborty, Fabio Pierazzi, and VS Subrahmanian. Ec2: Ensemble clustering and classification for predicting android malware families". In: IEEE Transactions on Dependable and Secure Computing (2017).

Fauzia Idrees Fauzia Idrees et al. Indroid: A novel Android malware recognition system using ensemble learning methods". In: Computers & Security 68 (2017), pp. 36{46.

Arvind Mahindru Arvind Mahindru and Paramvir Singh. Dynamic permissions based android malware recognition using machine learning techniques". In: Proceedings of the 10th innovations in Software Engineering Conference. 2017, pp. 202{210.

Zhenxiang Chen Zhenxiang Chen et al. Machine learning based mobile malware recognition using highly imbalanced network traffic". In: Information Sciences 433 (2018), pp. 346{364.

Md Shohel Rana Md Shohel Rana, Charan Gudla, and Andrew H Sung. Android malware recognition using stacked generalization". In: 27th International Conference on Software Engineering and Data Engineering. 2018.

Md Shohel Rana Md Shohel Rana, Sheikh Shah Mohammad Motiur Rahman, andAndrew H Sung. Evaluation of tree-based machine learning classifiers for android malware recognition". In: International Conference on Computational Collective Intelligence. Springer. 2018, pp. 377{385.

Suleiman Y Yerima	Suleiman Y Yerima and Sakir Sezer. Droid fusion: A novel multilevel classifier fusion approach for android malware recognition". In: IEEE transactions on cybernetics 49.2 (2018), pp. 453{466.
Hemalatha A	Hemalatha A and Selvabrunda. Mobile Malware Recognition usingAnomaly Based Machine Learning Classifier Techniques". In: International Journal of Innovative Technology and Exploring Engieering (IJITEE) 8 (2019).