Research Article

App Doppelganger Detection via Artificial Neural Network

Shivansh Shuktel, Meenakshi Kumari, Neha Pal & S. Babeetha
SRM Institute of Science and Technology, Ramapuram, India

Article History: Received: 11 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 23 May 2021

Abstract: Detecting repackaged applications is one of the most pressing issues in the Android universe. Several attackers typically find a good program, alter or insert malicious code into it, repackage it, and sell it in the online business sectors. They use code obfuscation tactics to discourage app cloning and repackaging. Good code protection is another tool for identifying repackaged Android applications that we suggest. The approach examines the similarity of Android applications based on the method call specifics of component classes that receive inferred intents. We offer a counting-based method for efficiently filtering the noisy layout that can lead to deviation. It can distinguish between the program with absolute and partial level similarities. Another benefit of our approach is precision. We suggest an artificial neural network technique for filtering the noisy layout that can induce variance. It can detect absolute and partial application-level similarities. We assess the clone recognition approach on a constant informational index and check the Review and Exactness, which is critical. Moreover, our discoveries show that our technique is inconceivably excellent and precise in identifying different kinds of clones to decide the level of similitude in Android applications.

Key words: Doppelganger, Package, Clone, Detection, Redundancy.

1. Introduction

Millions of apps download in a single day from the Google play store without knowing its authentication. In the era of internet evolution, the market of the mobile app is increasing very rapidly, with which it's giving a chance to malicious app developers and a hacker to attract millions of users and earn wholesome money. To overcome the problem of a clone app for a user, we develop the clone

app detection technique. Our technique is working on the principle of artificial neural networks (artificial replica of the brain). All of the previously discovered apps have attempted to identify app clones based on UI structure similarity. Both of those apps were ineffective at determining type 6 clones. When it comes to app tuning, our model can perform six types of cloning techniques, ranging from type 1 to type 6. Type 1 is the simplest app clone, while type 6 is the most complicated app clone. The earlier discovered applications used for app cloning could detect up to type two or a maximum of type three clones off applications. As a result, we can see that their solution was inefficient.

Additionally, they had to manually figure out the gaps in source code between the cloned application and the legitimate target application. As a result, productivity decreases. They did not incorporate machine learning, or as we call it, deep learning, into the methods, they used to identify clones.

2. Related Work

1.1 IOS/Android/Web App

i) IOS App

IOS stands for iPhone operating system. It is an operating system for iPhone,iPad, and other Apple devices. It was first released on 29 June 2007. Apple Inc developed it. It is used for IOS hardware. It is designed for smooth and easy networking for apple products. Its key features are accessibility, multitasking, task completion, Siri, etc. The word IOS is used to define Apple-based phones.IOS app can only be downloaded from its app store.

ii) Android-App

It is an operating system for Mobile phones. It consists of a modified version of the Linux kernel as well as much other open-source software. It is made for smartphones and tablets. It can be downloaded from any web store and app store. It was first released on 23 September 2008. It can be written in java,c, python, shell, etc. The android app developer is google, Huawei, Andy Rubin, Gameloft, etc.

iii) Web Application

It is an application that allows a web browser and web technologies to run any task using the internet. It is software that can be accessed from any place by only using a web browser. There are various web applications like multiple page app (Mpa), Single page app(Spa), dynamic web application, statics application, etc. The web application establishes the interaction between the application using database and middle wires. In this, many web applications can work simultaneously at a single point in time.

1.2 GUI or UI

GUI stands for graphical user interface, while UI stands for the user interface. In simple words, we can say that UI improves or establish interaction between user and electronic device. UI can be enhanced by using artificial intelligence. UI can be defined as a non-graphical interface. UI consists of two components output user interfaces and input user interface. The output user interface components are Led, screen, speaker, etc., while the input user interface is a mouse, keyboard, touch screen, etc.GUI is a subtype of UI. GUI comprises the graphical part that is taskbar, menu, and windows, etc.

1.3 Cloning of Apps

It is a process in which the copy of an app or website is done. We can access the composition of the app in two modes public and private. Cloning of app is not safe most of the time as operating system like Android is more dangers to malware or viruses. For finding out the doppelganger, we propose three methods.

i) Based on the Package

Third-party developer re-enginers the existing genuine packages and add malicious functionality to them. This helps them in achieving the natural and the similar functioning of that.

ii) Based on the Core Function

Basic functionality will be the same to attract the consumer other than that the third-party developer adds malicious functionality to that same package. When using the app, the consumer will not find any difference between the original app and the cloned app. After using it thoroughly, the consumer will be able to detect the differences.

iii) Based on Cloning of UI

For third-party developers, this is the easiest and efficient way of cloning apps and web pages. The UI structure will be the same as the original app to distract the consumer and perform malicious activity. Third-party developers try to develop the most relevant UI by adding appropriate attributes, like in the actual UI(in-app and web page), like responsiveness, fast loading, minimum space usage, etc.

3. Literature Survey

In 2014, an app was released to identify clones of legitimate apps common on the market. It was successful in detecting affluence both partially and fully. It works by comparing the source code of the legitimate app and the cloned app source code. It also noticed whether the cloned apps contained any malware that could damage the user. However, since the accuracy of the detection performed by this program was so low, it could not be considered a reliable application. It was also inefficient, so it was not recommended to be used frequently. When comparing the source codes of the cloned app and the legend app when they had the same tools, this app failed miserably.

In 2018, a similar application was introduced that promoted semantically comparable methods and was a little bit more effective in its arena. The focus of this proposed app was on the ways, and it's named in the source code that was written for this framework instead of relying on wind seeker fragments of the source code that were unnecessary and increased the overhead. As a result, this proposed system was efficient and accurate, with a high appreciation. However, when it came to larger datasets, it became confused. The prediction was very complex, so it could not provide an accurate value for the larger datasets for machine learning.

When the model was trained for a given set of data sets, the more comprehensive data set equation was very intrinsic, resulting in uncertainty. This device was insecure, with a security rating of two out of ten and a lengthy extraction period.

It was powerful enough to detect the cloned app without repackaging the cloned app in its entirety in a 2019 program. While not all of the APK files in the cloned application are packed, it was still adequate to a degree. Because of its technique when it came to holding off for the station, or we can say encryption of the code that is the load or told in the clone APK file, this application had the downside of not working out the clone apps very efficiently. It also lacked a hierarchical tree structure and displayed results that were not reliable or up to par.

Another application was proposed in the same year, and it was successful because it imposed similar features in the proposed framework. The end decryption was also more straightforward in this case, i.e., debugging was simple for this application to unpackaged or cloned apps, and it had a wide range of usable implementations. This application's major flaw was storing several values in a single variable, which generated an ambiguous situation and, as a result, complicated the application. Another disadvantage of this method was that it differed depending on the coding style used, which resulted in a high computation overhead, implying that it took a lot of time and money to develop. After that, this application was no longer able to provide reliable or approximate data. The results provided were far from what the user should have received in the first place.

Another application was proposed in 2019 to assure that it would have effective results when detecting clone apps that are as real as legitimate apps in the market with the same name and popularity. It enhanced the experience of the applications that were used to detect the. Finally, the app's reliability for detecting clones reduced the time it took to see clones while also lowering the cost of maintaining the app. However, it had flaws that made it a new Effective, such as payloads that were too high and large to carry. It became a bottleneck for this application in the end. It was complicated to get better performance from this application because the way it worked in its version could not be bettered, and it also did not have reliable results, so it eventually became useless. When it came to large datasets, the method became uncertain. The prediction became complicated, resulting in its inability to provide reliable results for larger datasets and unable to be further educated. This is another area where this application failed.

4. Proposed System & Methodology

Neural networks are a chain of ANN-based algorithms used to mimic or copy the Functions of the human brain to evaluate the relationship between the different and tremendous amount of data is brain consists of neurons. In the human brain, neurons can be used Transfer data from one part to another part of a brain, and a network of several millions of neurons can be called a neural network in artificial neural networks, each particular neuron can be classified as a node or a unit Which models the neurons in an artificial biological brain. Every artificial neural network consists of three components or three layers first one is the input layer, the second one is the hidden layer for the computational layer, and the third one of the last one is the output layer.

- Input Layer.
- Hidden (Computation) Layers.
- Output Layer.
    - The input layer takes every one of the info Signals and moves it to the following level. Then another capacity of the primary layer or the information layer is to assemble the data of its encompassing.
    - The computation layer plays out every one of the estimations that are going on behind the scenes
    - The output layer gathers all the data from the hidden layer and cycles it.

  Furthermore, the learning can be done in two steps forward propagation and backward propagation:
    - Forward propagation, it is nothing but a guess about the solution
    - Backward propagation is Used to minimize the error percentage between the solid solution and the guessed answer.

There are different types of artificial neural networks, each work on their own sets of rules. Let's see some of the famous neural network algorithms:-

1. Radial basis function neural network
2. RNN
3. Sequence to sequence model
4. Modular neural network
5. Feed-forward neural network
6. Convolutional neural network
7. Multilayer perceptron

In our proposed model, we used two types of algorithms, And they are as follows:

(i) Feed-Forward Neural Network:-

In this model of neural network in the node or unit moves in a single direction That is more forward in the order of the first layer than in the hidden nodes and at the last layer

(ii) Multilayer Perceptron

It is a feed-forward neural network that uses the backward propagation technique for the training of nodes. We are using this algorithm because of its predictive capability and hierarchy, or it's the multilayered structure of the networks.

5. Advantages of the Prospected System

- Non-Forceful Hardening Resilient
- Cutthroat Exactness
- Keeps up both exactness and productivity at a similar time
- Ensuring a reduced portrayal
- The prospected procedure can identify repercussed applications

6. Technologies Used

Backend Technologies

- Python
- Numpy
- Sci-learn
- Jupyter Notebook

Frontend Technologies

- Web Technologies
- Bootstrap

7. Architecture

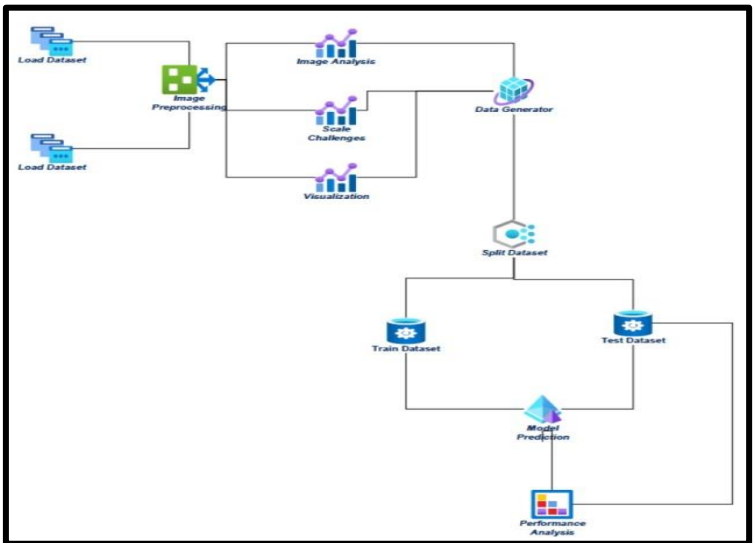

Figure 1: Architecture Diagram.

8. Modules

Exploratory Data Analysis

Exploratory data analysis is a Technique of scrutinizing data sets to rehash their peculiarity.

EDA Can be done typically By using two methods. The first one can be done via non-graphical form or can be done via the graphical method, And in the second method, the data can be univariate or multivariate(Mostly bivariate)

The graphical and non-graphical method non-graphical method deals with the Computation of demographic data sets.

On the other hand, the graphical method deals with the rehashing of datasets in visualizations and pictures. (Packing and estimation decline methodologies help make pictorial introductions of high-dimensional data containing various components.)

Univariate and multivariate methods:

Univariate method analysis one fickle (one column of the data set) at a time.

On the other hand, the multivariate method analyses three or more fickles at a time.

But in our proposed model, power multivariate will be just bivariate which means it exactly analyses two fickles at a time. Our model performs univariate analysis in each fickle of a multivariate before doing multivariate analysis.

In our model, we have gathered around 300000 datasets, so after doing univariate analysis and bi-/multivariate analysis, we have detected many deviant, flaky, and missing values, And also we have detected outliers (data sets that are not required) So as a result after removing outliers and finding the missing deals we have constructed the data set that is free of missing values and outliers. How about we take a model for a superior arrangement, assume we saw a criminal taking something from a precious stone store situated almost a lake? Is that lake behind a sanctuary? Since we are utilizing a neural network, approach our model Recreates cerebrum so as told in the model in the entire situation there is a robber there is a lake. There is a sanctuary here, lake, and the sanctuary is exceptions, and The robber is the necessary informational collection. Presently at whatever point our eyes see that robber again, our mind immediately remembers him so that We can make any move

Figure 2: Deployment.

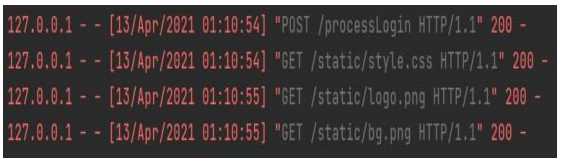

Figure 3: Log In.

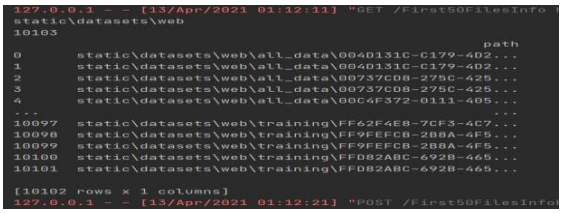

Figure 4: Calling First Fifty Sets of Data for Dataset Construction.

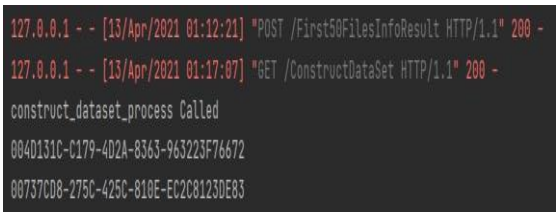

Figure 5.1

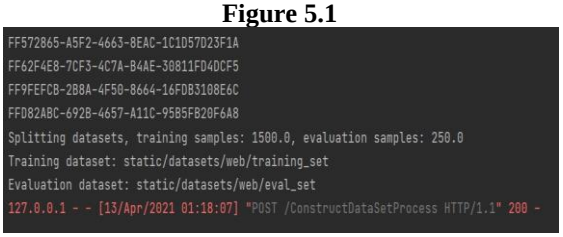

Figure: 5.2

Figure 5: Construction of Datasets for Training.

Feature extraction

We remove all method conjuring features from each section class that articulates assumption channels and assemble an identity.

We are chipping away at the location of the Application clone, and in our proposed model, one of the rules for the identification of clones by estimating the closeness of UI structure as we realize that an application comprises loads of UI components in which one of them is pictures so this method is utilized for our vision preprocessing and acknowledgement of conduct, designs, proficiency, and development .this highlights altogether characterize a picture. It is a simple image preprocessing technique that describes image patterns, so based on its output, we can say what we Identify on that image. For instance, a cross sign will be a component of a picture of a church. The main feature of this technique is to convert Visual Information into vector and raster form so that the information can be e extracted from the UI structures that we are using in our model, for example, radio buttons, buttons, text field, drop-down boxes, sliders, etc. this will help us in finding the cloned app based on the UI structure.

Figure 6: Preparing of Built datasets for Application Clone Expectation.

Data Augmentation

This procedure Is utilized to artificially support the size of the preparation dataset by generating new changed variations Of pictures in the informational index. For example, we can rotate an image in all directions. In our model, the proficiency is straightforwardly relative to the size of the informative index Means the more informational collections we train, the more effective we will get. The most well-known increase strategies utilized the following—wrap, evenness, reflection, Gaussian commotion, crop, flip, scale, interpretation, pivot. Our model uses this procedure because there are billions of applications existing today, so finding a cloned application with Fewer informational indexes won't deliver more productivity. That is why by extending the size of our informative index, we can undoubtedly accomplish A Striking proficiency.

Measuring Similarity

For detecting The cloned app, we have to measure the similarity between the apps. It can be done by differentiating the birthmark, package, functionality & UI structure between the two apps.

Our methodology depends on counterfeit neural organizations and the calculation that we are utilizing that is fed forward neural organization,

- In the first layer is the input layer. For the development of sources of info, we are taking the UI part (for example, a screen capture of the UI structure) as a JPEG/JPG/PNG, which is related to its own GUI record, after the preparation of the informational collections it will be changed over into NPZ format which is likewise connected with its own GUI variation. The transformation should be possible utilizing NumPy.

Figure7: Creation of Datasets.

- In the computation layer, we are Computing the similarity between the given apps within the derived parameters: app packaging, re-engineered code, functionality of the app, and user interface structure.
- In the output layer, it collects all the computed data and predicts whether the app is a clone or not.

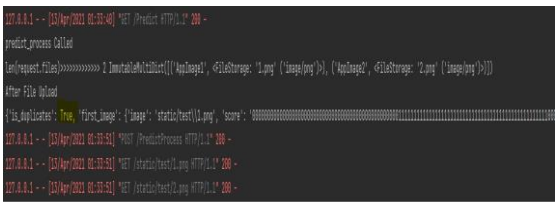

Figure 8: When App is Cloned (i.e., True)

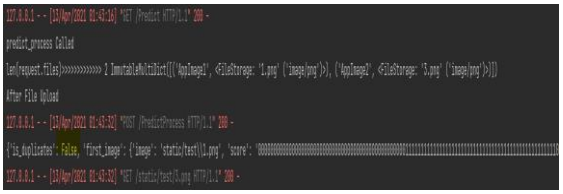

Figure 9: When App is Not Cloned (i.e., False)

9. Results

Datasets

To achieve clone detection, we need to get a massive amount of datasets to train our model. For that, we took leverage from some of the existing attempts. We used many real-time applications that are not officially verified and are present on various websites. we used

- Dataset from Apk Pure
- Dataset from Apk4Free
- Dataset from ihackedit.
- Dataset from Cracked Apk
- Dataset from ACMARKET
- Dataset from APKMB

Measuring Efficiency

The table given below shows the findings and results:

DATASET SOURCES	NO OF APPS	NO OF CLONE PAIRS	NO OF CLONED APPS PRESENT IN THE DATA SET	NO OF CLONED APPS DETECTED BY MODEL	EFFICIENCY(%)
Apk Pure.	42	18	12	9	75%
Apk4Free.	3961	2759	867	689	79.4694%
ihackedit.	237	167	55	32	58.1818%
Cracked Apk.	401	236	62	29	46.7741%
ACMARKET.	862	589	213	157	73.7089%
APKMB	39	25	6	6	100%
TOTAL→	5900	3794	1215	922	72.1890%

Figure 10: Table of Datasets.

For measuring efficiency, we created a data set of around 6000 apps that contain 3794 clone pairs. In total, we have examined 6000 apps from 6 data sets sources which include 3794 clone pairs. During primary research, we found out that out of 5900.

Apps in these six sources, around 1215 apps are cloned.

And after the training of the modules, our model can detect 922 cloned apps. So for this set of data, we have achieved an efficiency of 72.1890% ~73%.

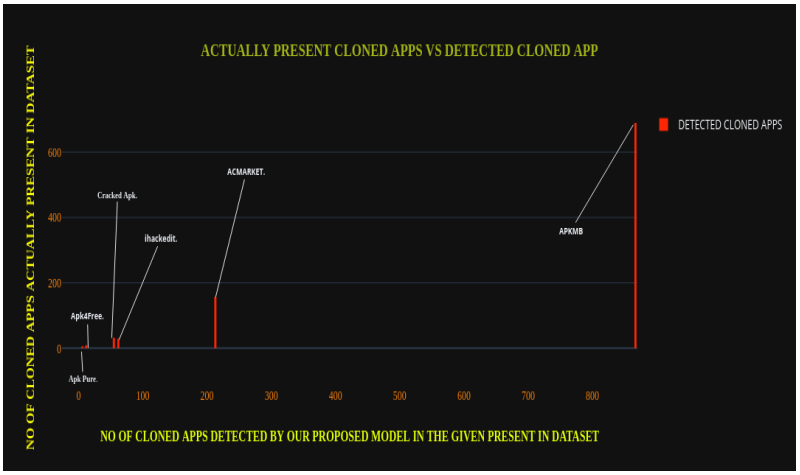

Figure 11: Graph Plotted between Actually Present Cloned Apps vs Detected Cloned App.

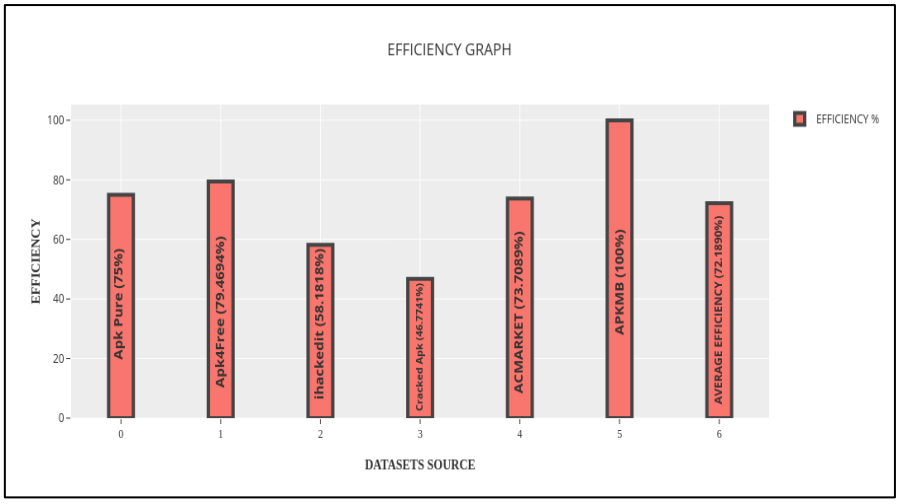

Figure 12: Efficiency Graph.

10. Implementation Screenshots

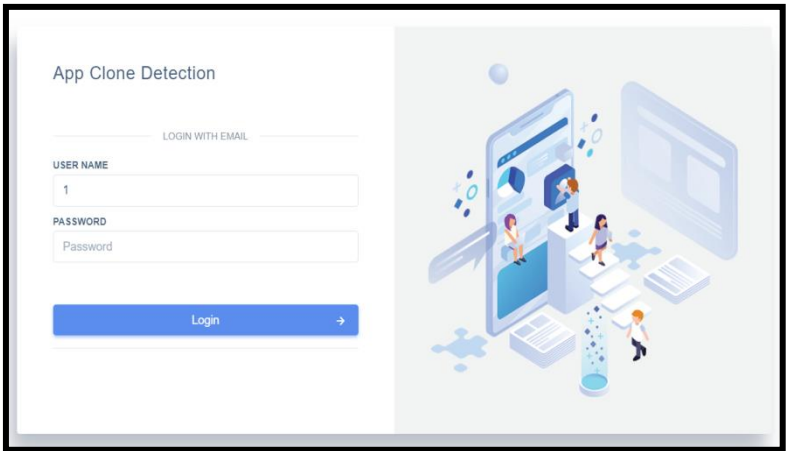

Figure 13: Login Page.

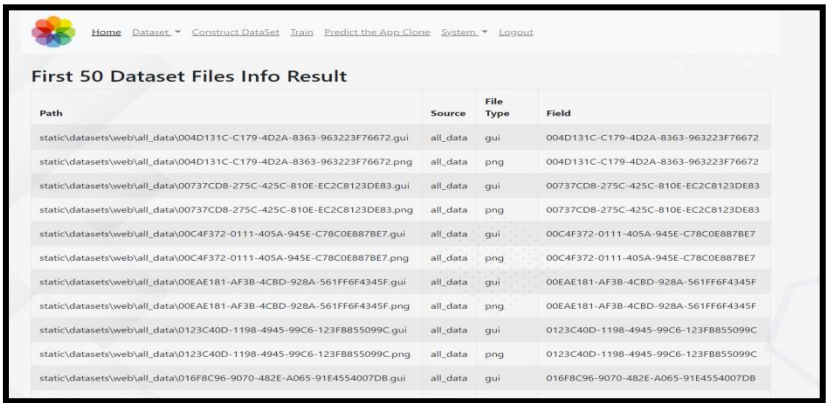

Figure 14: Dataset Gathering Page.

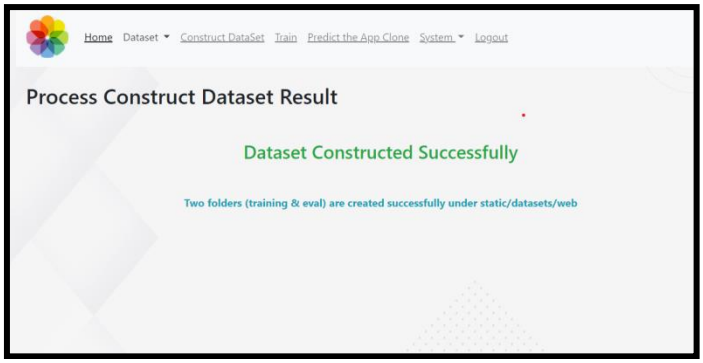

Figure 15: Dataset Construction Page.

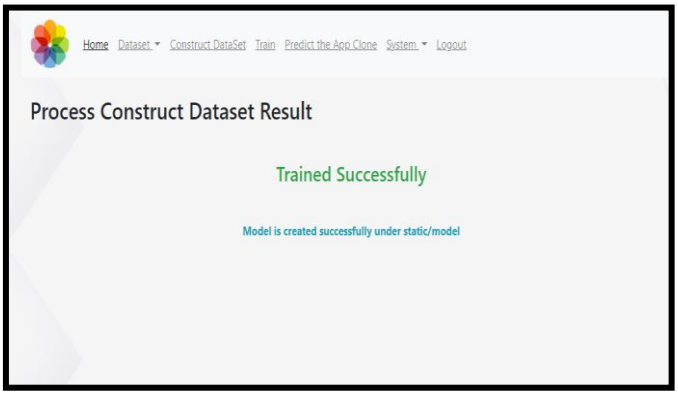

Figure 16: Dataset Training Page.

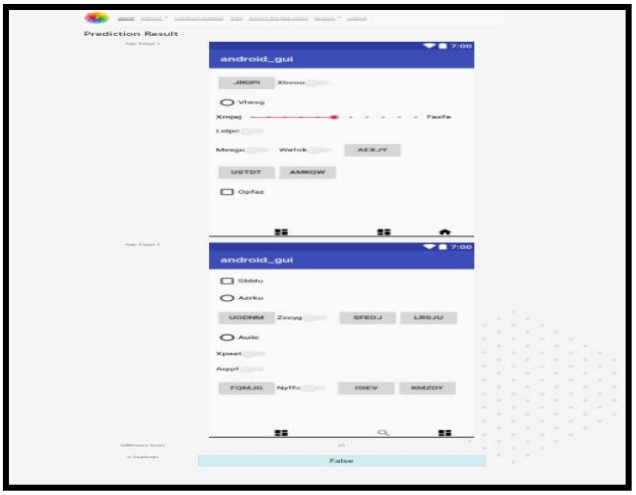

Figure 17: Output Page.

10. Conclusions

In this paper, we propose a novel app cloning/repackaging detection tool. For similarity measurement, it explores the method signature of an app like the existing detection tools. The unique feature of our proposal is that it considers only the component classes that receive implicit intents and extracts the method signature related to the courses at a small code. The assessment results show that this application can recognize application clones among the stuffed applications without bargaining the authentic application hardening. We likewise proposed an including-based method that can naturally and successfully channel the boisterous format without earlier information to improve precision.

11. Future Scope

In the future, for the spontaneous creation of the effectiveness given by the proposed model, the model Ought to have the option to Prepare more than 1.2 Lakh datasets at a time, as our present model trains 110697 datasets at that point, the model Ought to have the chance to take more than 50 datasets at a time for the location of anomalies and have the option to identify more exceptions since this will limit the repetition and eliminate the

information that isn't needed. The model should look at over two UI structures all at once because it will save time. The model ought to be store information of over 200 buyers all at once. The model ought to have the option to recognize malware and infections and fit for eliminating them. The frontend part can be upgraded later on, and endorsement security is additionally required.

Reference

1. John C.S.Lui, Mingshen Sun, Mengmeng Lui, "DroidEagle: Seamless detection of visually similar for Android apps," Security and Privacy in Wireless and Mobile Networks, 8th ACM Conference June 2015, pp.8-9.
2. Catherine A, A.Sumitha M.E, Aishwarya R" Boosting App Clone detection efficiency through robust approach neural network Application," Journal of Advanced Research in CSE and IT, March 2021, pp. 3-5.
3. Christopher Vendome, Michele Tufano, Martin white, Denys poshyvanyk "Deep learning code fragments for code clone detection," the 31st IEEE/ACMIC, August 2016, pp.12-15.
4. S. Niu, H. Niu and T. Yang, "Clone analysis and detection in Android applications," in Proc. 3rd International Conference System Information. (ICSAI), Nov. 2016, pp. 520–525.
5. Mitali S Mhatre, Dr Fauzia Siddiqui, Mugdha Dongre and Paramjit Thakur" Artificial Neural Network: A Prediction Technique" IJSER, Volume 6, Issue 12, December 2015, pp.2-3.
6. S. Capkun, L. Malisa and K. Kostiainen "Detecting mobile application spoofing attacks by leveraging user visual similarity perception," in Proc. 7th ACM Conference Data Application Security Privacy, Mar. 2017, pp. 290–295.
7. Qurat Ul Ain; Wasi Haider Butt; Muhammad Waseem Anwar; Farooque Azam and Bilal Maqbool" A Systematic Review on Code Clone Detection" in IEEE Volume 7,22 May 2019, pp.4-7.
8. Admane Ganesh Ulhas, "Malware Detection System and Classifications Techniques on Android" in IJARIIE, Vol-2, Issue-5,2016,pp-3.