Complex & Intelligent Systems (2020) 6:573–590
https://doi.org/10.1007/s40747-020-00154-3

ORIGINAL ARTICLE

Success prediction of android applications in a novel repository using neural networks

Mehrdad Razavi Dehkordi[1] · Habib Seifzadeh[1,2] · Ghassan Beydoun[3] · Mohammad H. Nadimi-Shahraki[1,2]

Received: 23 January 2018 / Accepted: 5 May 2020 / Published online: 3 June 2020

Abstract

Nowadays, Android applications play a major role in software industry. Therefore, having a system that can help companies predict the success probability of such applications would be useful. Thus far, numerous research works have been conducted to predict the success probability of desktop applications using a variety of machine learning techniques. However, since features of desktop programs are different from those of mobile applications, they are not applicable to mobile applications. To our knowledge, there has not been a repository or even a method to predict the success probability of Android applications so far. In this research, we introduce a repository composed of 100 successful and 100 unsuccessful apps of Android operating system in Google PlayStore™ including 34 features per application. Then, we use the repository to a neural network and other classification algorithms to predict the success probability. Finally, we compare the proposed method with the previous approaches based on the accuracy criterion. Experimental results show that the best accuracy which we achieved is 99.99%, which obtained when we used MLP and PCA, while the best accuracy achieved by the previous work in desktop platforms was 96%. However, the time complexity of the proposed approach is higher than previous methods, since the time complexities of NPR and MLP are $O(n^3)$ and $O(nph^koi)$, respectively.

Keywords Repository · Success and failure · Successful application · Failed application · Data set · Android

Introduction

Today, Android has a great share of smart-phone operating systems. Based on IDC statistics,[1] Android holds 86.8% of the market share in the early 2016Q3. Because of the large share of this operating system, diverse applications in different categories are available for it, most of which have attracted users and succeeded, while some have failed to do so.

Moreover, due to the development of many applications in various areas such as mobile phones, personal computers, and webpage development, their maintenance has become particularly important. Since numerous people use these applications and pages in different fields and applications and web pages with similar features are found in all areas, there is a great competition to attract users to each application. The success or failure of an app is related to its maintenance process. Therefore, developers must maintain the apps based on users' needs to prevent them from being discontinued or decommissioned.

The success or failure prediction of an application would not only compensate the damage, but also improve the maintenance process of the application and thus results in the use of more brand new features in an application by the developer to attract more users, adaptation of the app to the new environment, change of the user interface of application, improvement in the user's experience of running the

[1] http://www.idc.com/promo/smartphone-market-share/os.

✉ Habib Seifzadeh
seifzadeh@iaun.ac.ir

Mehrdad Razavi Dehkordi
mehrdad.razavi68@sco.iaun.ac.ir

Ghassan Beydoun
Ghassan.Beydoun@uts.edu.au

Mohammad H. Nadimi-Shahraki
nadimi@ieee.org

[1] Faculty of Computer Engineering, Najafabad Branch, Islamic Azad University, Najafabad, Iran

[2] Big Bata Research Center, Najafabad Branch, Islamic Azad University, Najafabad, Iran

[3] School of Systems, Management and Leadership, University of Technology Sydney, Sydney, Australia

application, and generally improvement in their application [1].

Software maintenance is a general process of changing a system after it has been delivered, e.g., correcting code errors, correcting design errors, and accommodating new requirements after release [2].

Thus far, using machine learning and artificial intelligence techniques, some works have been done on the failure or success of desktop computer applications. However, since the features that can be calculated for desktop computer applications are not calculable or available for mobile applications and there is no repository of successful and failed applications of Android operating system, so far nothing has been conducted on predicting the failure or success of mobile applications.

Repositories are central storage files, in which a mass of data is stored.[2] They can be used, for instance, in version-control systems[3] and software component reuse [3].

A successful project is one which is completed on time and on budget, with all features and functions originally specified. On the other hand, a failed project is a project which is completed and operational, but is over-budget, over the time estimate and with fewer features and functions than initially specified, or a project which is canceled before completion or not implemented at all [4].

A discontinued app is defined here as: a failed, retired, and decommissioned app on Google Play, for which no update has been released in 2 years or more, or has been removed from the store for security, financial, personal, or other reasons and the user encounters the "Not Found" error while searching the name of its package on Google Play.

Application retirement is a disposal process which aims to end the existence of the software. It includes the total removal of the system or retaining a number of required parts [5]. Application decommissioning is the process of removing a system, application, database, or platform from service while keeping its important data [6].

Artificial intelligence (AI) is concerned with building computer systems that solve the problem intelligently by emulating the human brain [7].

Artificial neural networks, commonly referred to as "neural networks," are massively parallel-distributed processors made up of simple processing units (neurons), which perform computations and store knowledge [7].

Since no research has been performed so far in predicting the failure or success of applications of Android operating system, in the present study, a repository of 100 successful and 100 unsuccessful applications of Android operating system on the Google Play Store with 34 features for each application was built. Then, the data were given for training to neural networks of LVQ and MLP with different learning functions and NPR. The accuracy of the above-mentioned neural networks was examined by applying the PCA algorithm and selecting the effective features and not applying the PCA algorithm with 34 features provided in the prediction.

The main contribution of this paper is expressed as follows:

1. Others have done so much work about the success or failure prediction of desktop applications, but nothing has been done on the success and failure prediction of the applications in Android operating system.
2. There exists no repository of successful and unsuccessful applications in Android operating system. Due to the need of such repository in prediction, a repository of 100 successful and 100 unsuccessful applications has been made.
3. Many scholars have used the preexisting available databases or those made by applications for the purpose of prediction, but in this study, the made repository of successful and unsuccessful applications (including: discontinued, retired, and decommissioned applications from the Google Play Store™) has been produced and provided by the authors.
4. Due to the fact that the features used in the success and failure prediction of desktop computers applications are not available for the Android operating system applications or at best cannot be calculated, to predict, new features will be extracted which can be calculated for all applications of Android operating system.
5. For each app in the repository, 34 features have been extracted through counselling with scholars and other college students, using Questionnaires and ISO 25010 Standard of software quality.
6. By calculating the correlation coefficient for all the obtained features, it appears that all the extracted features are needed for the prediction and none of them shows dependence to each other and also all features are of the equal significance in comparison with others.
7. The made repository is sent to different machine learning techniques to train and make a model. Then, the accuracy parameter of each technique will be calculated and the comparison of all techniques with one another will be followed.

This paper is structured as follows: in Sect. 1, we present some fundamental definitions. In Sect. 2, some explanations are provided about neural networks. In Sect. 3, we review the studies on repository development and the works performed about predicting the success or failure of applications. Section 4 provides the research methodology followed during this investigation. Section 5 specifies the experimental

[2] http://searchoracle.techtarget.com/definition/repository.

[3] https://techterms.com/definition/repository.

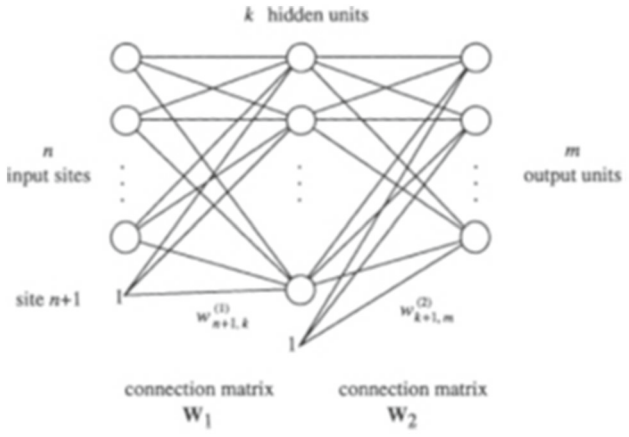


Fig. 1 Structure of neural networks [18]

results. Finally, in Sect. 6, conclusions and suggestions for improving the repository and prediction are offered.

Background

A. Artificial neural networks
Artificial neural networks, commonly referred to as "neural networks," are massively parallel-distributed processors made up of simple processing units (neurons), which perform computations and store knowledge [7]. Modelling a complex system by artificial neural networks (ANNs) gives the possibility to fully take nonlinearities into account [8].
The use of neural networks in this study for prediction has four underlying justifications: first, the neural networks are those which are data-driven and self-adaptive and have a tendency to reduce errors and adapt themselves to the model; second, they can be generalized and deduce the unobserved data (new) even with noise; third, they are non-linear [9]; fourth, they can be learned with small data size [10]; and finally, artificial neural networks can be used in predicting software defects, quality, and risks [11–14] estimating the expended efforts of an application [15], estimating the level of Fault Injection in the developmental process of an application [16], and estimating the modules expenses [17] .

B. MLP (Multi-layer perceptron)
In Fig. 1, the structure of an MLP neural network is shown. Consider a neural network with n inputs, k hidden units, and m outputs. As shown in Fig. 1, the weight between input unit i and hidden unit j is w_{ij}^1, and the weight between hidden unit i and output unit j is w_{ij}^2. In the following figure, the weight between constant 1 and hidden unit j is shown by $w_{(n+1,j)}^1$, and the weight between constant 1 and output unit j is $w_{(n+1,j)}^2$.
W1 is the weight matrix for side 1 (between the input unit and middle unit) and W2 is the weight matrix for side 2 (between the middle unit and output unit). In neural networks, the network compares the output value after training in each step with the expected output, calculates the error after computing the difference between the network output and expected output, and makes the network output approach the expected output as much as possible by changing the weights [18].

C. LVQ (Learning vector quantization)
This neural network is developed for a group of algorithms, which are widely used in the large-scale categories and data. The successful application of the LVQ algorithm has been confirmed in difficult problems such as medical data analysis, fault detection in technical systems, and satellite data spectrum categorization. This algorithm is highly useful for researchers and people working outside the machine learning field who are looking for quick categorization methods. The categorization in the LVQ algorithm is based on the calculation of distance, that is, the calculation of the Euclidean distance, in which the similarity of received data to primary data or classes is calculated [19].

D. NPR (Neural pattern recognition)
The term "pattern recognition" covers a wide range of information processing problems including speech recognition, handwritten letter categorization, and error detection in machine and medical systems [10]. The traditional pattern recognition methods need the correct position of the input pattern, but in practice, it is shown that the input pattern is often displaced, rotated, resized, or becomes incomprehensible for the network due to noise. As a result, the traditional pattern recognition methods might face difficulties in recognition in such conditions. Today, pattern recognition methods are introduced using neural networks with no sensitivity to rotation, resizing, noises, or input pattern displacement [20].

E. SVM (Support vector machine)
Support vector machines represents a new statistical technique that has drawn much attention in recent years. SVM are based on the structural risk minimization (SRM) induction principle, which aims to restrict the generalization error (rather than the mean square error) to certain defined bounds [21]. SVM have recently been applied to a range of problems that include pattern recognition, bioinformatics, and text categorization [22].

F. kNN (K-nearest neighbors)
The kNN algorithm is one of the simplest and most commonly used machine learning methods. The method is an important approach to nonparametric classification and is quite easy and efficient. Its rule of classification assigns

a class label for unknown samples by estimating its *k*-nearest neighbors based on known samples [23].

G. Random forest
Random Forests is the unique learning machine that has no need of an explicit test sample because of its use of bootstrap sampling for every tree. This ensures that every tree in the forest is built on about 63% of the available data, leaving the remaining approximately 37% for testing the OOB (out-of-bag) data [24].

H. Decision tree
Decision trees are sequential models, which logically combine a sequence of simple tests; each test compares a numeric attribute against a threshold value or a nominal attribute against a set of possible values. The logical rules followed by a decision tree are much easier to interpret than the numeric weights of the connections between the nodes in a neural network. The Decision Tree algorithm attempts to generalize, or find patterns in, the data. It does so by determining which tests (questions) best divide the data into separate classes, forming a tree [25].

I. Principle component analysis (PCA)
Principal component analysis is a technique for linearly mapping multidimensional data onto lower dimensions with minimal loss of information. Principal component analysis has been extensively applied in almost every discipline, chemistry, biology, engineering, meteorology, etc [26].

Previous research

Numerous studies have been conducted on the development of different repositories, the most important of which are: Providing a source code repository for competitive programming between students [27]; Designing and implementing an open-source software repository known as OpenCom in Shanghai Component Library to describe, store, retrieve, collect, and develop open-source software and assist programmers in component-based programming [28]; introducing a reusable component repository in embedded systems, in which components data are stored in XML documents [29]; proposing a new source code repository for dynamic storage, browsing, and retrieval of source codes, in which software is divided into software units including application >module >class >function >sub-function [30]; and introducing a repository of Android open-source applications including 4416 versions of 1179 applications through the mining of an Android open-source repository called F-Droid and sharing it on a personal website [31].

All these studies are summarized in Table 1. Accordingly, numerous repositories of reusable components and open-source software in the domain of desktop-computer software, Android, and software source codes have been developed so far. Nevertheless, no repository of successful and failed Android apps is available.

Many studies have been conducted in predicting the failure or success of desktop computer applications and ERP (enterprise resource planning) software and webpages, the most important of which include: dynamic prediction of project success with completed design percentage parameters, cost of the owner, cost of the project contractor's obligations, cost of project owner's obligations, hours of efforts of the project owner, exact working period, cost of order change, amount of order change and non-working days due to bad weather using genetic algorithms, fuzzy logic, and neural network by Chien-Ho Ko et al. (2007) [7]; prediction of project success using a combination of support vector machine algorithms with the fmGA genetic algorithm by the completed design percentage parameters, cost of the owner, cost of factor construction, hours of effort considered by the designer, material form and tools, cost of construction, cost of the project owner's obligations, and incidence rate measured by Min-Yuan Cheng et al. (2008) [22]; prediction of the success of application using dependency rules learning using the risk-taking parameters of the project, including shortage of personnel, unrealistic planning and scheduling, development of wrong functions and properties, development of wrong user interface, bad project management, etc. by Xiaohong Shan et al. [4] (2009); prediction of software shortcomings using the RFC, LOC, CBO, WMC, LCOM, DIT, NOC, LOCQ, and NOD parameters using Bayesian Networks by Ahmet Okutan et al. (2014) [32]; and prediction of project failure factors and use of these factors in predicting failure and success using unrealistic expectation parameters of the project, lack of a board of directors and leader for the project, inadequate obligations of stakeholders, inadequate project management and control, requirements changes, unclear requirements, etc., by Logistic Regression Method by Guillamue et al. (2015) [33].

In Table 2, all the works done in predicting the failure and success of applications in all the fields are mentioned. Accordingly, nothing has been done in predicting the failure and success of the applications of Android operating system.

Research methodology

As mentioned before, the necessity of a repository of successful and unsuccessful Android applications is due to two reasons. First, there is no such repository, and second, no work has been carried out to predict the failure and success of Android applications so far.

In repositories which have been used by others in the success or failure prediction, software defects, and effective factors in the failure of the desktop apps, Procaccino et al. could obtained 42 projects through using questionnaire

Table 1 Things done in the development of Repository

Years	Authors	Description of the study	Repository holding
2003	Lee et al.	Designing and developing a repository of reusable components [34]	N/A
2005	Dorta et al.	Developing a source code repository called OpenScr to assist the OpenMP community by being used in shared memory multiprocessor platforms [35]	A repository containing software source codes in all languages
2007	Burgio	Designing and implementing a reusable repository and explaining the required features for its development [3]	Totally, 423 classes are divided into 83 packages, including 19367 line of codes
2008	Punitha et al.	Introducing a source-code repository for competitive programming and evaluating the skills of students [27]	Source codes in different languages to be used in competitive programming by students
2009	Hong Min et al.	An open-source software repository called OpenCom in Shanghai Component Library to promote the use of open-source software in China	A RAS-based repository of open-source software
2009	Bajracharya et al.	Proposing an infrastructure for accessing , showing and analyzing large open-source codes called Sourcerer [36]	A large number of open-source codes stored using a meta-model
2011	Chang et al.	Introducing a repository of reusable components in embedded systems which store components data in XML documents [29]	A repository of reusable components with features such as editing, searching, and retrieving the components, as well as providing solutions to remove useless components
2013	Chakraborty et al.	Proposing a source code repository for dynamic storage, browsing, and retrieval of source codes [30]	A source code repository, in which each software is divided from the smallest to the largest software unit
2015	Krutz	Introducing repository of open-source apps through mining the open-source Android app called F-Droid [31]	A repository of open-source apps including 4416 versions of 1179 apps

[23]. Wohlin et al. have used 46 projects in the prediction, which were obtained through using NASA-SEL database [24]. Chien-Ho Ko et al. have used 54 existing projects in CAPP database and have chosen 15 projects for evaluation [7]. Min-Yuan Cheng et al. have used 46 projects available in CAPP database [22]. Xiaohong Shan et al. have used the 100 projects made by the Project Data Generator application [4]. Francisco Reyes et al. could extract 140 projects for prediction through using questionnaire, whose database contained 104 successful and 34 unsuccessful projects [37]. Ramaswamy et al. have used the 12 chosen projects for software defect prediction of application [38]. Sumanv have used

Table 2 Things done in predicting success or failure of applications

Authors	Features	Method	Prediction case	Platform
Procaccino et al. (2002)	Requirements, management, users and customers, estimation and planning, project manager, software development process etc. [23]	Logistic regression	Success and failure of project	Desktop computers
Claes Wohlin et al. (2005)	Delivery in due time, software quality and proper maintenance [24]	Analysis of key success stimuli	Success or failure of similar projects	IBM computers
Chien-Ho Ko et al. (2007)	Percentage of completed design, cost of owner, cost of project contractor's obligations, cost of project owner's obligations etc. [6]	Combining genetic algorithm, fuzzy logic and neural network as well as introducing a new method, known as EPSPM	Dynamic prediction of project success	CAPP projects
Min-Yuan Cheng et al. (2008)	Percentage of completed design, cost of owner, cost of factor construction, hours etc. [17]	Combining support vector machine with FmGA genetic algorithm Providing a new method, known as ESIM	Project success	CAPP projects
Xiaohong Shan et al. (2009)	Project risk-taking including shortage of staff, unrealistic planning and scheduling, development of wrong methods and properties, development of wrong user interface, bad project management, continuous flow of change in requirements, etc. [4]	Association rule mining	Software success by obtained rules	Desktop computers
Francisco Reyes et al. (2011)	Project output [37]	Genetic algorithm for cost optimization	Cost of developing different software parts	Desktop computers
Ramaswamy et al. (2012)	Software defects [38]	K-Means clustering algorithm with random forest classifier algorithm	Success of software	ERP and web domain applications
Suma et al. (2014)	Software Defects [24]	Random forest classifier algorithm	Software Defects	Desktop computers
Ahmet Okutan et al. (2014)	RFC, LOC, CBO, WMC, LCOM, DIT, NOC, LOCQ and NOD [32]	Bayesian networks	Software Defects	Open-source applications in web domain

Table 2 continued

Authors	Features	Method	Prediction case	Platform
Guillamue et al. (2015)	Factors affecting failure including unrealistic expectations of project, lack of board of directors and leader for project, inadequate obligations of stakeholders, inadequate project management and control etc. [33]	Logistic regression	Project failure factors and use of these factors in predicting failure and success	Large enterprise applications including tourism industry, tax administration, airport management
Guillamue et al. (2015)	Factors affecting failure and helping the companies in big decisions and adopting policies when the project is failing [39]	Logistic regression	Project failure factors	Projects of information technology management companies in the field of desktop computers

15 projects for the success and failure prediction of applications [24]. Ahmet Okutan et al. have used public databases such as Ant, Tomcat, Jedit, Velocity, Synapse, Poi, Lucene, Xalan, and Ivy, which velocity with 229 samples had the lowest number of samples and Tomcat with 858 sample containing open-source applications had the highest number of samples [32], and 105 unsuccessful projects were extracted by Guillamue et al. and were used to predict the success and failure of the project [33,39].

According to the fact that in works done by others, the numbers of projects in the prediction were different.

Moreover, in Table 2, used features by others are different and there is no standard for the necessary number of projects used in prediction and number of features. In this article, a repository of 100 successful and 100 unsuccessful applications including 34 features from the applications of Google Play Store was built. To do so, the successful applications were extracted from the Top Free and Top New sections on the Google Play Store, and unsuccessful applications including retired, discontinued, and decommissioned applications, were extracted from resources such as alternative.to, etc., such that there was at least one application for each category on the Google Play Store. Then, the repository was given to MLP and LVQ neural networks with different learning functions and NPR for prediction, and the accuracy of the above-mentioned neural networks was examined via employing PCA algorithm and selecting effective features, without applying the PCA algorithm with 34 features provided in the prediction.

A. Features of Proposed Repository

The proposed repository included the successful and failed apps on Google Play Store. The features for each app in this repository were extracted based on the ISO 25010 software quality survey and standard.[4] Each app in this repository had the following features: (1) Name, (2) Number of packages, (3) Number of classes, (4) Number of Methods, (5) LOC for each app, (6) Developing country, (7) Apk file size, (8) Having a top developer or not, (9) In-app purchase option, (10) Having a sponsor, (11) Relation with the social media networks, (12) Google Play category, (13) Being open-source, (14) In-app advertisements, (15) Needing root access, (16) Being free, (17) Having a name relevant to the operation, (18) Performance efficiency, (19) Relation with system apps, (20) Interoperability, (21) Usability, (22) Accessibility, (23) Reliability, (24) Install accountability, (25) Having documentation, (26) Showing permissions while installing the app, (27) Having support, (28) Having a database for storing information, (29) Having cloud storage for storing information, (30) Having one or multiple developers, (31) Having a forum or blog, (32) Having a help or FAQ (frequently asked questions) menu, (33) Having a voting option on the Google Play Store, (34) Having a donation option, and (35) Success or failure of the app. In Table 3, the extraction of each feature is explained.

B. Development stages of the proposed repository

Fig. 2 illustrates the general process of developing the repository.

First, using the "discontinued apps" section on Alternative.to[5] and other sources, all the discontinued, retired, and decommissioned apps were identified and a list of apps was prepared in separate folders. Then, the pres-

[4] https://nl.wikipedia.org/wiki/ISO_25010.

[5] http://alternativeto.net/tag/discontinued/?platform=android&sort=likes.

Table 3 Extraction of features

Feature	Extraction method
Number of packages In APK	Consulting the supervisor
Number of classes In APK	Consulting the supervisor
Number of all methods in APK	consulting the supervisor
LOC	Review articles
Country	Application PlayStore page
APK size	Application PlayStore page
Top developer	Application PlayStore page
InApp purchases	Application PlayStore page
Sponsored	Consulting the supervisor
Relation with social networks	Review articles
Category	Application PlayStore page
Open source	Review articles
Contain adds	Program failure due to excessive advertising
Needs root	Application PlayStore page
Free	Application PlayStore page
Name relation with app operation	Functional suitability in ISO 25010 standard
Performance efficiency	Resource utilization a sub-section of performance efficiency, a category of ISO 25010 standard
Relation with system apps	interchangeability (compatibility) a category of ISO 25010 Standard
Accessibility (install on different android OS?)	Adaptability a sub-section of portability, category of ISO 25010 standard
Install accountability (install newer version on old version)	Replaceability a sub-section of portability, a category of ISO 25010 standard
Having documentation	From review articles
Show permissions when installing app	Security from ISO 25010 standard
Support (having email for bug report)	Quality in use from ISO 25010 standard
Programs has database?	Consulting the student of another university
Have one developer or more?	Application PlayStore page
App has a forum or blog	Quality in use from ISO 25010 standard
App has help or FAQ?	Quality in use from ISO 25010 standard
Has voting option (rate for APP)	Quality in use from ISO 25010 standard
App has donate option	Quality in use from ISO 25010 standard
Discontinued	From alternative to and other sources
Accessibility (install on different android OS?)	Adaptability a sub-section of portability, a category of ISO 25010 standard

ence of capture for each app on Internet Archive[6] was checked; in case no capture was available, another app would be selected. In the next step, in case capture was available, each app's page on Google Play would be saved and features 1, 6, 7, 8, 9, 12, 14, 15, 16, 25, 27, 30, and 31 would be extracted. To extract feature 12, we used Table 4. Afterwards, the classes.dex file was extracted from each app's apk file. We employed the dex2jar tool to convert the classes.dex file into a .jar file for analyzing the app in Cyvis.

Cyvis[7] is the software used for analyzing the complexity and viewing the classes, functions, etc. in .jar files. In the next step, the number of methods and lines of code for each app was counted using the dex-methods-count[8] and Swing LOC Counter tools, respectively. Afterwards, each app was installed on a Huawei Honor 3C H30-U10 device with Android 4.4.2 and H30-U10V100R001C900B310 build number, and features 11, 13, 14, 19, 20, 22, 24, 26,

[6] http://archive.org/web/.

[7] http://cyvis.sourceforge.net/.

[8] https://github.com/mihaip/dex-method-counts.

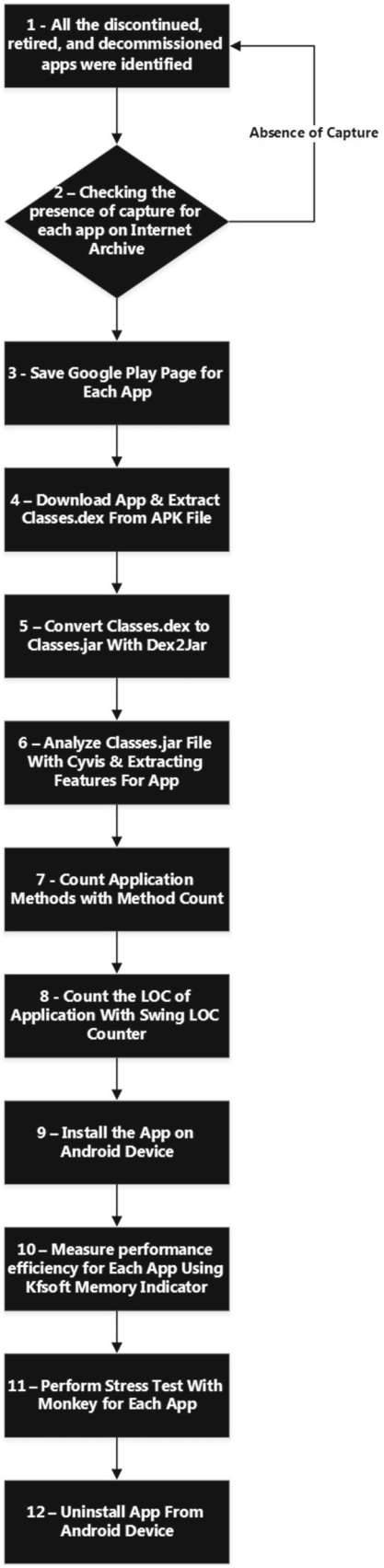


Fig. 2 Steps of developing the repository and extracting the features

Table 4 Converting the app categorization into numbers

Number	Category
1	Android Wear
2	Art and design
3	Auto and vehicles
4	Beauty
5	Books and reference
6	Business
7	Comics
8	Communication
9	Dating
10	Education
11	Entertainment
12	Events
13	Finance
14	Food and drink
15	Health and fitness
16	House and home
17	Libraries and demo
18	Life style
19	Maps and navigation
20	Medical
21	Music and audio
22	News and magazine
23	Parenting
24	Personalization
25	Photography
26	Productivity
27	Shopping
28	Social
29	Sports
30	Tools
31	Travel and local
32	Video players and editors
33	Weather
34	Games

28, 29, 32, 33, and 34 were extracted.

In the following step, the performance efficiency of the apps (including RAM and CPU usage) was measured using the Resource Monitor Mini[9] tool and converted into numerical data by Table 5. Then, we used the Monkey tool to perform stress tests five times on each app to evaluate the reliability feature. Monkey[10] enters a set of random events including touch, rotation, silencing, tak-

[9] https://play.google.com/store/apps/details?id=info.kfsoft.android.memoryindicator.

[10] https://www.utest.com/articles/how-to-stress-test-your-android-app-with-monkey.

Table 5 Parameters for artificial neural networks algorithms

Method	Error	Parameters	Max fail	Epochs
MLP with trainlm learning function	MSE	Three hidden layers, first layer including 5 neurons, second layer including 5 neurons, and third layer including 2 neurons	6	1000
MLP with trainbr learning function	MSE	Three hidden layers, first layer including 10 neurons, second layer including 5 neurons, and third layer including 2 neurons	6	1000
LVQ with learnlv1 learning function	MSE	One hidden layer including 10 neurons	6	50
LVQ with learnlv2 learning function	MSE	One hidden layer including 10 neurons	6	The learning process will be conducted two times: first time 50 and the second time 100
NPR	MSE	One hidden layer including 75 neurons	6	500

Table 6 Parameters for kNN, SVM, RF, and DT algorithms

Method	Error	Parameters
kNN	MSE	10 Fold cross validation and 100 neighbors
SVM	MSE	10 Fold cross validation
Random forest	MSE	10 Weak Learners
Decision tree	MSE	100 Fold cross validation

ing screenshots, playing music, and turning Wi-Fi on/off, as the input into the app.

We would enter 1 for the reliability feature if an app successfully completed the test and 0 if we received the "Force Close" message during the test. Finally, all the apps were uninstalled from the Android device.

Since most features used in this repository were not numerical data, we employed different criteria for converting each feature. Table 6 presents these criteria and the method for their conversion into analyzable numbers.

– The usability feature is a subjective criterion. Therefore, for its conversion into numbers, we turned to 7 individuals who had most frequently visited Android software service centers. All the 200 applications were given to 7 individuals. They were asked to assign 1 in case of satisfaction with user interface and 0 otherwise. The total votes were rounded up.
– A few categories were added to Google Play after 2015 and, thus, did not include any failed apps.
– In the proposed repository, there is at least one app for each Google Play category.

C. Data analysis and sharing

The resulting repository was shared on GitHub[11] website to be used in data analyses and prediction of the success or failure of Android apps. For example, by inspecting the "Developing Country" column, we concluded that the success or failure of apps did not correlate with the developing country. By checking the "Reliability" column, almost all successful apps were found to pass the stress test successfully, while most of the failed apps encountered problems during the test.

The proposed repository consisted of 200 (100 successful and 100 failed) apps. The Clash of Clans game and VOA Farsi had the largest (66 mb) and smallest (1.1 mb) apk file sizes, respectively, among the successful apps. Moreover, Advanced English Vocab and Foxifi Addon had the largest (25 mb) and smallest (0.0.2 mb) apk file sizes among the failed apps, respectively. In addition, 39% of the failed apps did not complete the Stress test, while only 4% of the successful ones failed.

In statistics, the frequency (or absolute frequency) of an event i is the number n_i of times that the event occurs in an experiment or study.[12]

Figures 3 and 4 illustrate the absolute frequency of successful and failed apps in Google Play categories.

Based on Fig. 3, we concluded for instance that the repository held 6 successful apps belonging to Google Play

[11] https://github.com/mehrdadr68/Android-Successful-Failed-Apps-Repository.

[12] https://en.wikipedia.org/wiki/Frequency_(statistics).

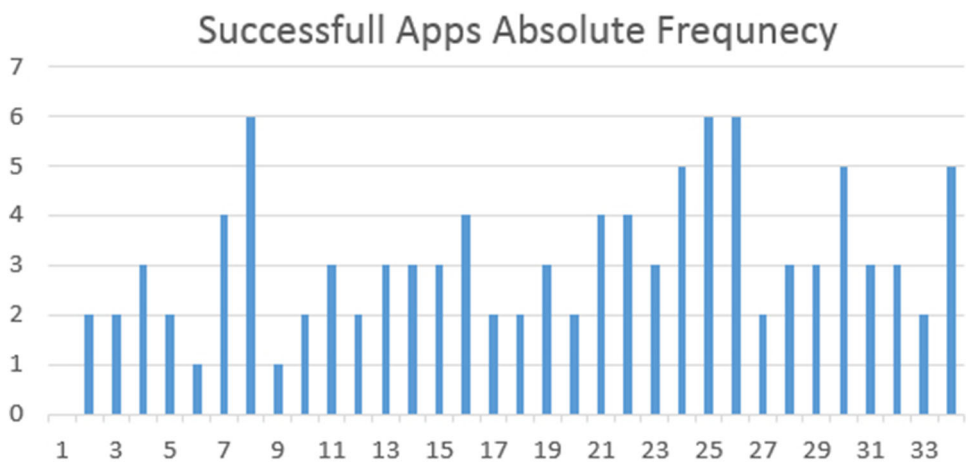


Fig. 3 Absolute frequency of successful apps in each category

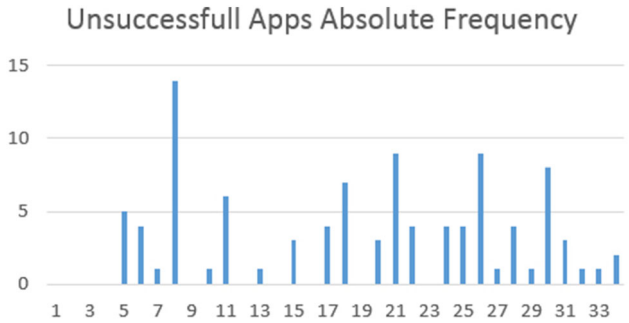


Fig. 4 Absolute frequency of failed apps in each category

category 8. Also, we concluded from Fig. 4 that the repository had 9 failed apps belonging to Google Play category 21.

Figure 5 demonstrates the percentage of failed apps for each category in the form of a pie chart. Based on this chart, the most failed apps belonged to category 8, i.e., "Communication".

The reason is that many developers have been producing apps to attract users, but have failed, because they could not compete with other apps in the Communication category.

Figure 6 depicts the percentage of successful apps for each category in the pie chart. Based on the above chart, most apps belonged to Google Play categories 25 and 26, i.e., Photography and Productivity, respectively.

Based on the "Donation Option" column, we concluded that almost none of the successful apps had this option in any menu, while the failed apps included it, because old apps did not have advertisements. However, nowadays, developers include them to regain the development costs.

To examine the dependence or lack of dependence of each of the features obtained in the repository, Chi-squared test or the correlation coefficient can be used.

Chi-squared test: It is used in two cases: a. lack of dependence between two variables in a table to evaluate the level of input dependency and brand quality, family population, TV size, educational background, occupation, etc., b: goodness-of-fit test to measure the difference between the frequency distribution of observed variable with expected frequency distribution. Since there is no expected frequency distribution for the features mentioned in the repository, Chi-squared test cannot be used to measure the dependency of the features [40].

Correlation coefficient: In statistical terms, correlation is the evaluation of a linear two-way relationship between two continuous variables. According to statistics, the correlation coefficient is a number between -1 and 0, which means that the negative relationship between two variables or might be a number between 0 and 1, which indicates the positive relationship between two variables. The correlation in statistics is expressed and calculated with the correlation coefficient. The correlation coefficient is calculated in two ways:

A: Pearson's correlation coefficient: is used when the two variables have a normal distribution; B: Spearman's correlation coefficient: is used which the two variables have an unbalanced distribution or ordinal data [41].

To calculate the correlation coefficient among the features mentioned for each program in the repository, MATLAB software was used. Since this software automatically performs the normalization of data to calculate the correlation coefficient, Pearson's correlation coefficient was employed.

To obtain the effective features in the prediction, the PCA algorithm was used to reduce features. After applying the PCA algorithm on 35 obtained features, 16 features were selected for prediction. Then, to find whether these features depended on each other and whether or not a

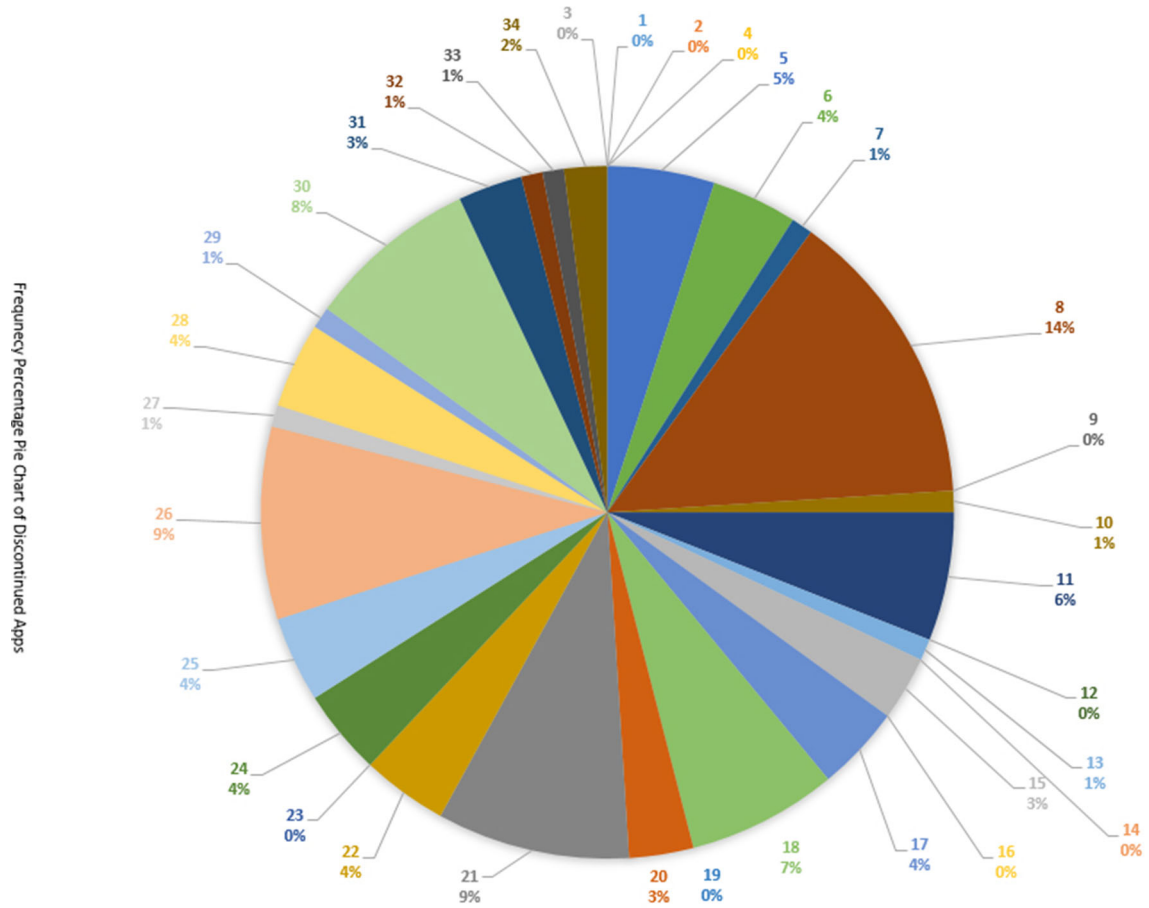


Fig. 5 Pie chat of percentages of failed apps in each Google Play category

feature could be calculated using another, Pearson's correlation coefficient was calculated for the remaining 16 features. The minimum correlation coefficient was equal to −0.2098 and the maximum was 0.5050. Since the highest and lowest correlation coefficients were not −1 and 1, it can be concluded that all 16 selected features were required in the prediction, and none of them can be calculated from another with no direct or indirect dependence.

As depicted in Fig. 7, in Phase 1, (1) first, the input comprising 34 features (33 features for prediction and 1 including the label of the class the software belongs to) is given by MATLAB as the input to the neural network of a categorization algorithm; (2) Then, the data are divided by the categorization algorithm, and 70% of the data are selected for training, 15% for validation, and 15% for testing. (4) For each algorithm, ten iterations of tests (training, validation, and testing) are performed, and in each iteration, the level of accuracy of the algorithms is calculated. (5) Finally, the output of this phase is given to Phase 3.

In Fig. 8, in Phase 2, (1) First, the input containing 34 features is given by MATLAB as the input to the PCA algorithm; (2) Then, the output of the PCA algorithm which is a dataset with 16 features is given to the categorization algorithm; (3) next, the data are divided by the categorization algorithm, and 70% are selected for training, 15% for validation, and 15% for testing. (4) For each algorithm, ten iterations of tests (training, validation, and testing) are performed, and in each iteration, the level of accuracy of the algorithms is calculated. (5) Finally, the output of this phase is given to Phase 3.

In Fig. 9, it is clear that the output of Phases 1 and 2, including accuracy in all 10 iterations of implementing the selected categorization algorithm, is given to Phase 3, so that the best algorithms for categorization of successful and unsuccessful programs are identified with and without applying the PCA algorithm.

Moreover, due to the fact that in other work conducted on predicting success or failure of applications by others, none of the features used by them has priority over the others. Therefore, none of the features used in this study and also in the conducted experiments through the use of made repository takes priority over the other and all features are of equal priority in the prediction.

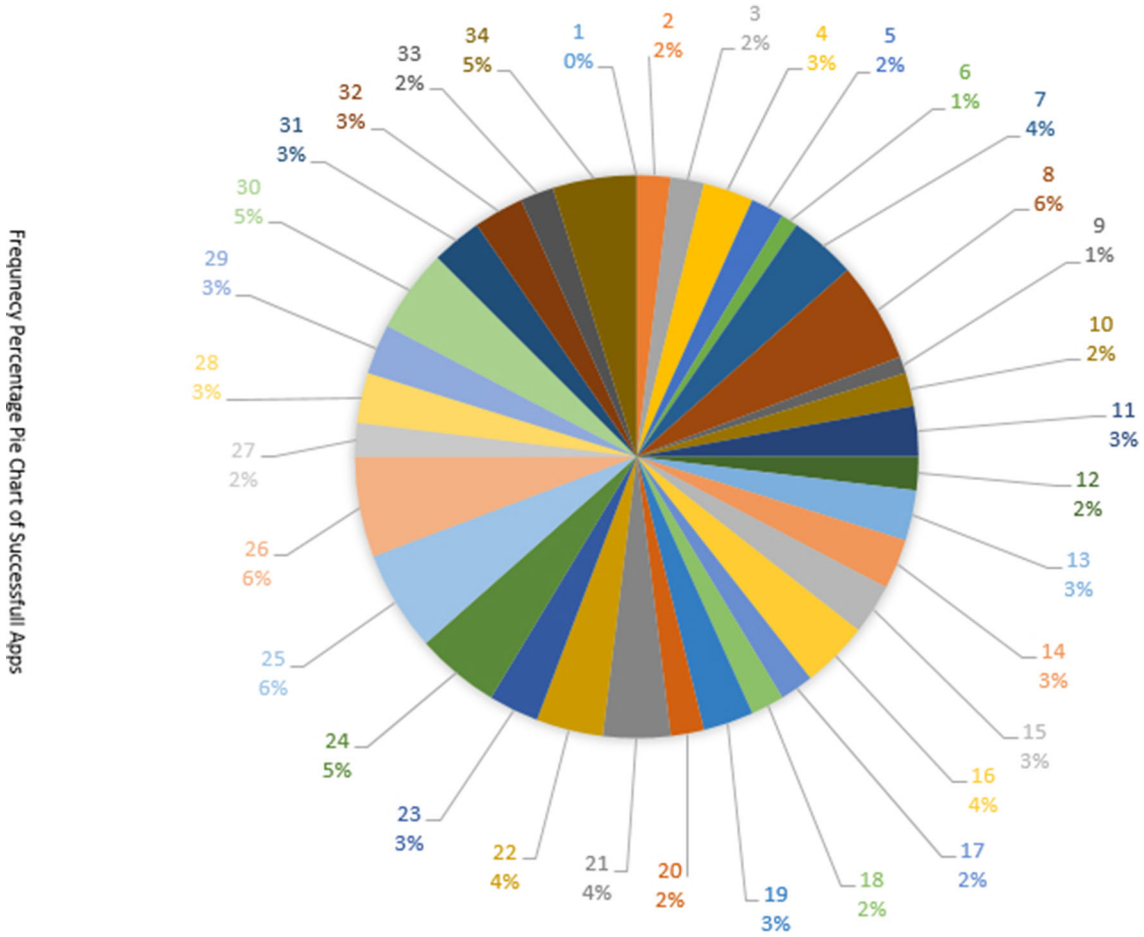


Fig. 6 Pie chat of percentages of successful apps in each Google Play category

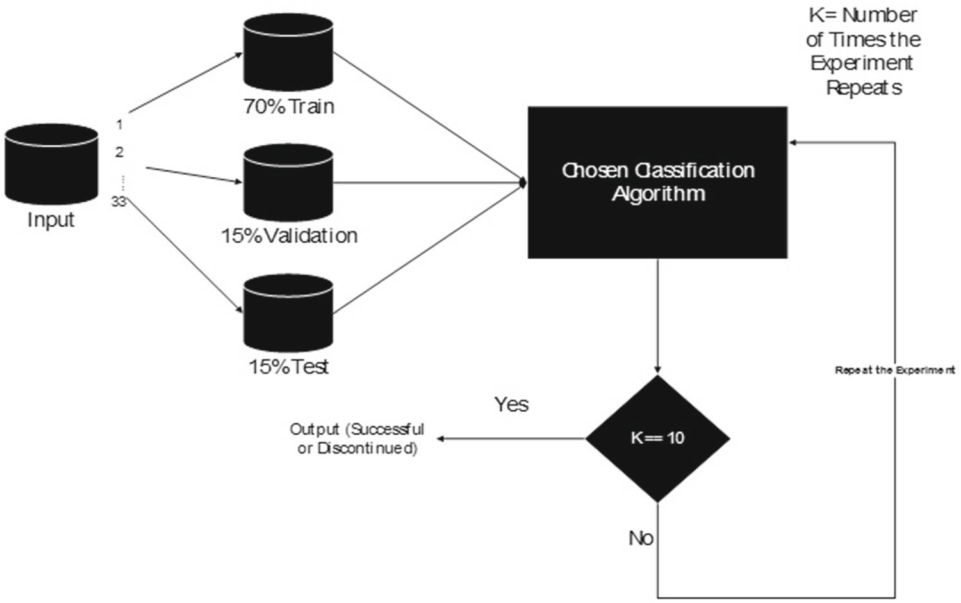


Fig. 7 Phase 1—steps to perform experiments using the created repository without applying PCA

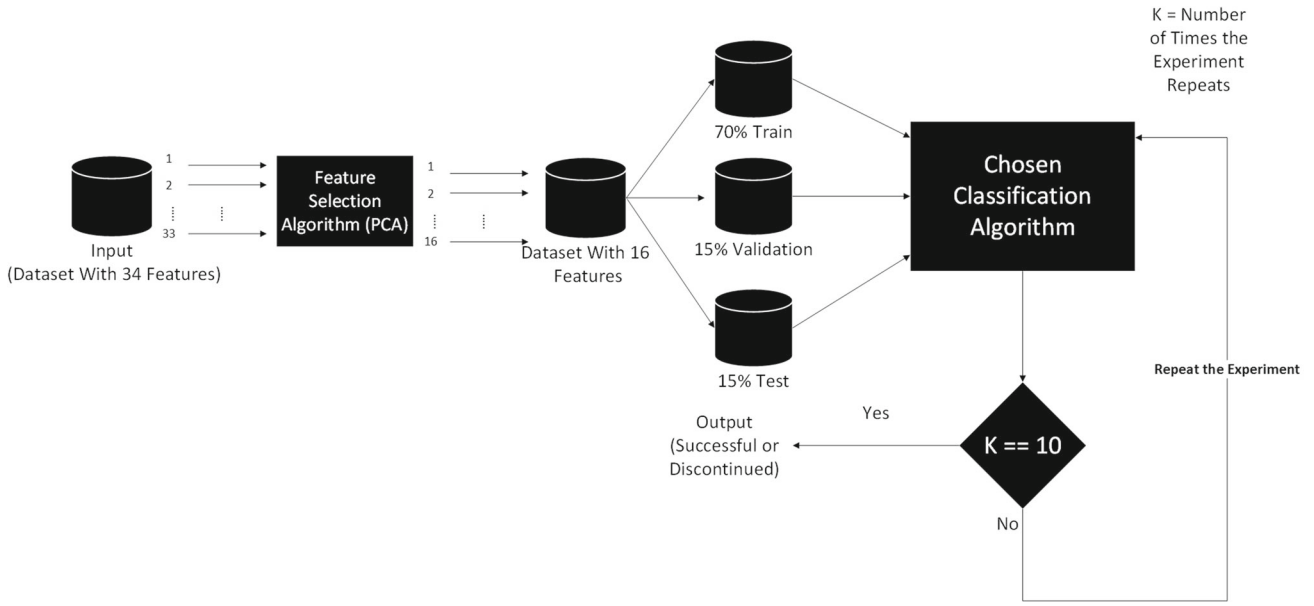
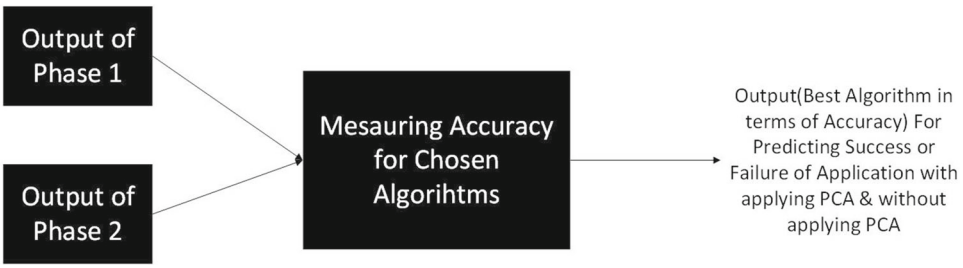


Fig. 8 Phase 2—steps to perform experiments using the created repository with applying PCA

Fig. 9 Phase 3—the output of Phases 1 and 2 are given to Phase 3 to measure accuracy and select best algorithm for classification



Experiment results

A. Parameter tuning
In Tables 5 and 6, all details of used neural networks in prediction such as MLP LVQ NPR neural networks and other sorting algorithm like KNN SVM Random Forest and Decision Tree have been mentioned.

B. Experiments environment
All tests were performed on an AMD A4 5300B computer with 8Gb CPU and running Windows 7, using MATLAB version R2017b.

C. Experiments
After building the desired repository, it was given to five neural networks: MLP with trainlm learning function; MLP with trainbr learning function; LVQ with leranlv1 learning function; LVQ with lerarnlv2 learning function; and NPR and also the desired repository was given to kNN, SVM, random forest, and decision tree algorithms. The accuracy parameter for neural networks and kNN, SVM, Random forest, and decision tree algorithms was calculated separately for each algorithm using the PCA algorithm to reduce the features and obtain the effective features and without PCA algorithm. For each algorithm, the training process was implemented ten times and the results of the highest accuracy for each algorithm were measured and shown in Table 7. To train the neural network and test whether the neural network maintained the pattern or not, 70% of the data were selected for training, 15% for validation, and the remaining 15% for testing. Moreover, the runtime for each of the five neural networked with/without the PCA algorithm has been calculated in seconds. Each neural network and learning algorithm was run ten times, and mean runtime for each is given in Table 8.

And in Table 9, the time complexities of the proposed algorithm are compared with the previous ones found in the literature. The use of the PCA algorithm for reducing the number of features in many cases will increase the precision. As it can be seen in Figs. 10 and 11, for example in MLP algorithm, the train, test, and validation error have been reduced after using PCA in comparison with no PCA algorithm used on features.

The criterion of comparing our work with others in different platforms is the accuracy in prediction. As mentioned

Table 7 Best accuracy for each algorithm

Algorithm	Accuracy	
	With applying PCA	Without applying PCA
MLP with **trainlm** Learning function	99.99%	89.47%
MLP with **trainbr** Learning function	99.99%	75.68%
LVQ with **learnlv1** Learning function	76.5%	77.5%
LVQ with **learnlv2** Learning function	76.5%	77%
NPR	87%	95.5%
kNN	76%	76%
SVM	85%	88.5%
Random forest	74.6%	76.85%
Decision tree	85.5%	86%

Table 8 Average runtime for each algorithm

Algorithm	Runtime	
	With applying PCA	Without applying PCA
MLP with **trainlm** Learning Function	5.3 s	1.798 s
MLP with **trainbr** Learning Function	2.392 s	3.281 s
LVQ with **learnlv1** Learning Function	14.39 s	10.995 s
LVQ with **learnlv2** Learning Function	18.696 s	18.654 s
NPR	1.572 s	1.572 s
kNN	9.168 s	10.07 s
SVM	2.01 s	1.258 s
Random forest	1.986 s	2.138 s
Decision tree	19.081 s	17.766 s

in Sect. 4, we use the accuracy as our main comparison criterion for all algorithms (with or without PCA). This is because it has also been used by the previous approaches for predicting success of applications in other platforms [14,22,24,32,33,42].

In Table 7, the use of the PCA algorithm lowered the prediction accuracy in some cases, which demonstrates that the use of PCA algorithm is not always the appropriate way to reduce features.

This is because not only PCA does not remove features in some cases, but also it discovers and utilizes features' relationships if possible. In other words, PCA may reach a composite feature that leads to the accuracy decrement in those circumstances. For example, we achieved the accuracy of 95.5% by leveraging the sole NPR algorithm, while applying PCA to it caused the accuracy to decrease to 87%. Hence, it is not always appropriate to use PCA to reduce the number of features. However, applying PCA to MLP algorithm always performed well and achieved 99.99% accuracy in prediction.

The notations used in Table 9 are as follows: n represents training samples, p stands for features, k is hidden layers each of which contains h neurons, and o represents output neurons. In addition, i is the number of iterations, $n_{treesis}$ the number of trees (for methods based on various trees), n_{sv} is the number of support vectors, and n_{l_i} stands for the number of neurons at layer i in a neural network.

Table 9 Computational complexity of applied algorithms

Algorithm	Computational complexity	
	Training	Prediction
MLP [43]	–	$O(nph^k oi)$
LVQ [44]	–	$O(n^2 logn)$
NPR [45]	–	$O(n^3)$
kNN [46]	–	$O(np)$
SVM [46]	$O(n^2 p + n^3)$	$O(n_{sv} p)$
Random forest [46]	$O(n^2 pn_{\text{trees}})$	$O(pn_{\text{trees}})$
Decision tree [46]	$O(n^2 p)$	$O(p)$
RNeural network [46]	?	$O(pn_{l_1} + n_{l_1} n_{l_2} + \cdots)$
Naïve Bayes [46]	$O(np)$	$O(p)$

As shown in Table 9, Naïve Bayse and KNN algorithms used in MLP training function have better time complexities. However, they fail to show high accuracy in prediction. Therefore, it can be concluded that an algorithm with low computational complexity may not have good prediction accuracy.

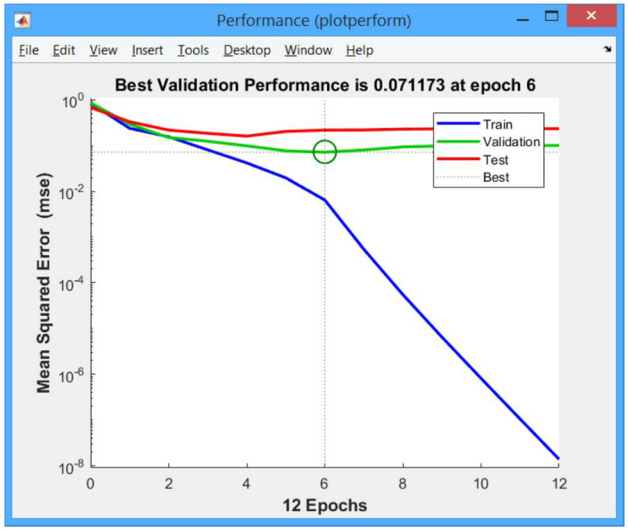


Fig. 10 Mean-squared error for train, test, and validation in MLP before applying PCA algorithm

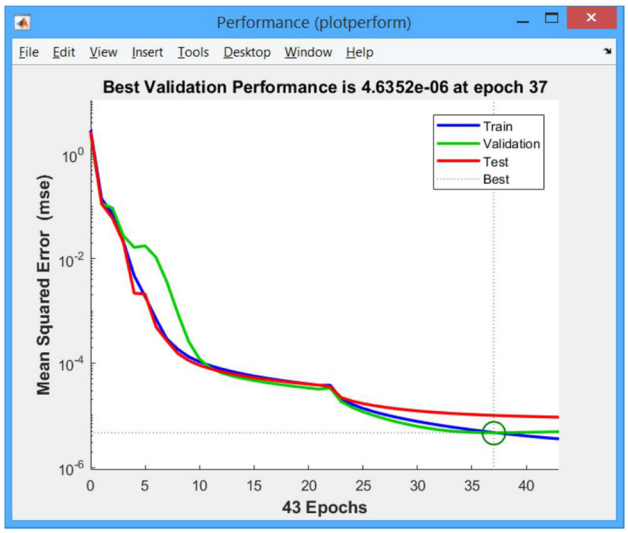


Fig. 11 Mean-squared error for train, test, and validation in MLP after applying PCA algorithm

It is also worth mentioning that since NPR and MLP algorithms used in the proposed approach have time complexities of $O(n^3)$ and $O(nph^koi)$, respectively, the computational complexity of our approach is high; and hence, it is not a good practice to utilize it online.
In Table 10, the accuracies of previous success prediction methods in desktop and web platforms are compared with the accuracy of the proposed approach.

As it can be seen in Table 10, the highest accuracy obtained in the desktop platform is 96%, while the highest accuracy achieved in Android operating system is 99.99%, which pertains to the proposed approach. This means that the accuracy achieved by us in the field of Android applications is greater than the accuracy achieved by others in the Desktop and Web platforms.

Table 10 Accuracy comparison of previous approaches with the proposed method

Authors	Prediction case	Platform	Accuracy
Cheng and Wu [22]	Project success	Desktop	45.3%
Reyes, Cerpa et al. [42]	Cost of developing different software parts	Desktop	81.45%
Suma et al. [24]	Success of software	Desktop	86%
Fenton et al. [14]	Software defects	Desktop	93%
Okutan and Yıldız [32]	Software defects	Open-source applications in web domain	95%
Guillaume-Joseph and Wasek [33]	Project outcomes and Success	Desktop	96%
Our case study with applying PCA	**Success or failure of android applications**	**Mobile**	**99.99%**

Conclusion and suggestions

As there has been no repository of successful and unsuccessful applications in Android operating system, a repository of successful and unsuccessful applications was built with 35 features.

Furthermore, this repository can help experiments and assist programmers develop and maintain better and more successful apps. The features included for each app in this repository can be used for future analyses. We can determine the importance of each feature in the analyses by examining the features of successful and failed apps.

The repository was then given to various neural networks & other classification algorithms for prediction to investigate the accuracy of each algorithms in two cases of with and without applying PCA algorithm. As in Sect. 5, the NPR algorithm had prediction with the accuracy of 95.5% without applying PCA, and MLP algorithm had prediction with the accuracy of 99.99% with applying PCA. This means that when there is a high number of repository features, the NPR algorithm is more accurate. Also, in the case the effective

features are extracted using the PCA algorithm, the MLP algorithm is more accurate.

As mentioned in Sect. 5, we used accuracy as our comparison criterion as it had been by other approaches, as well. Then, the proposed method was compared with the other approaches in terms of performance, and it was clear that the most accurate algorithm is not necessarily the most efficient one.

Therefore, it is concluded that MLP in combination with PCA should be leveraged if the accuracy is important, while Naive Bayse or KNN must be employed if the time is the main concern. It was also mentioned in Sect. 5 that the accuracy achieved by the proposed method is higher than those of previous approaches in desktop and web platforms.

In the future, researchers can design a tool for examining the usability feature so as to automatically check the number of menus, buttons, and other elements in apps and enter appropriate values for this feature. Also, researchers can use other tools such as Neoload to perform more accurate Stress tests.

In the prediction phase, other techniques of machine learning can be used in prediction. The compiled repository is shared on GitHub.[13]

References

1. Seifzadeh Habib, Abolhassani Hassan, Moshkenani Mohsen Sadighi (2013) A survey of dynamic software updating. J Softw Evolut Process 25(5):535–568
2. SOFTWARE ENGINEERING (2011) 9th edn. Addison-Wesley,
3. Buregio VA, Almeida ES, Lucredio D, Meira SL (2007) Specification, design and implementation of a reuse repository. Proc Int Comput Softw Appl Conf 1(Buregio VA, Almeida ES, Lucredio D, Meira SL. Specification, design and implementation of a reuse repository.):579–582
4. Shan X, Jiang G, Huang T (2009) A framework of estimating software project success potential based on association rule mining. In: 2009 International Conference on Management and Service Science, pages 1–4, Sept 2009
5. Kajko-Mattsson M, Fredriksson R, Hauzenberger A (2014) EM3: Software Retirement Process Model
6. IBM. Application retirement: enterprise data management strategies for decommissioning projects, 2008
7. Ko C-H, Cheng M-Y (2007) Dynamic prediction of project success using artificial intelligence. J Construct Eng Manag 133(4):316–324
8. Pasini A (2015) Artificial neural networks for small dataset analysis. J Thorac Dis 7(5):953–960
9. Zhang G, Patuwo BE, Hu MY (1998) Forecasting with artificial neural networks. Int J Forecast 14(1):35–62
10. Hippert HS, Pedreira CE, Souza RC (2001) Neural networks for short-term load forecasting: a review and evaluation. IEEE Trans Power Syst 16(1):44–55
11. Zhang D (2003) applying machine learning algorithm in software development
12. Mie M, Thwin T, Tong-Seng Q (2002) Application of neural network for predicting software development faults using object-oriented design metrics. In: Proceedings of the 9th International Conference on Neural Information Processing, vol 5, pp 2312–2316
13. Amasaki S, Takagi Y, Mizuno O, Kikuno T (2003) A Bayesian belief network for assessing the likelihood of fault content. In: Proceedings - International Symposium on Software Reliability Engineering, ISSRE, 2003-Janua, pp 215–226
14. Fenton N, Neil M, Marsh W, Hearty P, Marquez David, Krause Paul, Mishra Rajat (2007) Predicting software defects in varying development lifecycles using Bayesian nets. Inform Softw Technol 49(1):32–43
15. Bibi S(2015) Software process modeling with Bayesian belief networks. (December)
16. Pérez-Miñana E, Gras J-J (2006) Improving fault prediction using Bayesian networks for the development of embedded software applications. Softw Test Verifi Reliab 16(3):157–174
17. Dejaeger K, Verbraken T, Baesens B (2013) Toward comprehensible software fault prediction models using bayesian network classifiers
18. Rojas R (1996) Neural networks: a systematic introduction
19. Biehl M, Ghosh A, Hammer B (2007) Dynamics and generalization ability of LVQ algorithms. J Mach Learn Res 8:323–360
20. Reid MB, Spirkovska L, Ochoa E (1989) Rapid training of higher-order neural networks for invariant pattern recognition. Int Jt Conf Neural Netw 1:689–692
21. Huang Cheng Lung, Wang Chieh Jen (2006) A GA-based feature selection and parameters optimizationfor support vector machines. Expert Syst Appl 31(2):231–240
22. Cheng M-Y, Wu Y-W (2008) Dynamic prediction of project success using evolutionary support vector machine inference model. In: The 25th International Symposium on Automation and Robotics in Construction. ISARC-2008, pages 452–458, Vilnius, Lithuania, jun 2008. Vilnius Gediminas Technical University Publishing House Technika
23. Li L, Zhang Y, Zhao Y (2008) K-Nearest Neighbors for automated classification of celestial objects. Sci China Ser G Phys Mech Astron 51(7):916–922
24. Suma V, Pushphavathi TP, Ramaswamy V (2014) An Approach to Predict Software Project Success Based on Random Forest Classifier. In Suresh Chandra Satapathy, P. S. Avadhani, Siba K. Udgata, and Sadasivuni Lakshminarayana, editors. In: ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of CSI - Volume 2, volume 249 of *Advances in Intelligent Systems and Computing*, pages 329–336. Springer International Publishing, Cham
25. Kotsiantis SB (2013) Decision trees: a recent overview. Artif Intell Rev 39(4):261–283

[13] https://github.com/mehrdadr68/Android-Successful-Or-Failed-Apps-Repository.

26. Dong D, McAvoy TJ (1996) Nonlinear principal component analysis—based on principal curves and neural networks. Comput Chem Eng 20(1):65–78
27. Punitha T, Pradeep Kumar R, Seshadri R, Srinivasan R (2008) Mooshak—a valuable repository of codes. In: Proceedings - The 8th IEEE International Conference on Advanced Learning Technologies, ICALT 2008, pp 644–646
28. Ren HM, Yan ZY, Zhang JZ (2009) Design and implementation of RAS-based open source software repository. In: 6th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2009, 2:219–223
29. Chang CH, Lu CW, Chu WC, Yang CT, Hsiung PA, Hsueh NL, Koong CS, Shao KK (2011) XML-based reusable component repository for embedded software. In: Proceedings - International Computer Software and Applications Conference, pp 345–350
30. Chakraborty PR, Chowdhury AK, Chowdhury S, Hasan SA (2013) A new source code repository for dynamic storing, browsing, and retrieval of source codes. In: 2013 international conference on informatics, electronics and vision, ICIEV 2013, pp 0–5
31. Krutz DE, Mirakhorli M, Malachowsky SA, Ruiz A, Peterson J, Filipski A, Smith J (2015) A dataset of open-source android applications. In:2015 IEEE/ACM 12th Working Conference on Mining Software Repositories, volume 2015-August, pp 522–525. IEEE, may 2015
32. Okutan A, Yildiz OT (2014) Software defect prediction using Bayesian networks. Empir Softw Eng 19(1):154–181
33. Guillaume-Joseph G, Wasek JS (2015) Improving software project outcomes through predictive analytics: part 1. IEEE Eng Manag Rev 43(3):26–38
34. Lee J, Kim J, Shin GS (2003) Facilitating reuse of software components using repository technology. In: Proceedings-Asia-Pacific Software Engineering Conference, APSEC, 2003-Janua, pp 136–142
35. Dorta AJ, Rodriguez C, De Sande F, González-Escribano A (2005) The OpenMP source code repository. In: Proceedings-13th Euromicro Conference on Parallel, Distributed and Network-Based Processing 2005, PDP 2005, 2005 (October 2016), pp 244–250
36. Bajracharya S, Ossher J, Lopes C (2009) Sourcerer: an internet-scale software repository. In: Proceedings-International Conference on Software Engineering, pages 1–4
37. Reyes F, Cerpa N, Candia-Véjar A, Bardeen M (2011) The optimization of success probability for software projects using genetic algorithms. J Syst Softw 84(5):775–785
38. Suma V, Pushpavathi TP, Ramaswamy V (2012) An approach to predict software project success by data mining clustering. In: International Conference on Data Mining and Computer Engineering (ICDMCE'2012), page 6
39. Guillaume-Joseph G, Wasek JS (2015) Improving software project outcomes through predictive analytics: part 2. IEEE Eng Manag Rev 43(3):39–49
40. Verma JP (2013) Chi-square test and its application. In: Data Analysis in Management with SPSS Software. Springer India, India, pages 69–101
41. Mukaka MM (2012) Statistics corner: a guide to appropriate use of correlation coefficient in medical research. Malawi Med J 24(3):69–71
42. 1.17. neural network models (supervised)– scikit-learn 0.22.2 documentation, Mar 2020. [Online; Accessed: 08-Mar-2020]
43. Grbovic M, Vucetic S (2009) Learning vector quantization with adaptive prototype addition and removal. In: Proceedings of the 2009 International Joint Conference on Neural Networks, IJCNN'09. IEEE Press, page 911-918
44. Xu C, Chen S, Cheng J (2015) Network user interest pattern mining based on entropy clustering algorithm. In *2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, pages 200–204
45. Computational complexity of machine learning algorithms— the kernel trip, Mar 2020. [Online; Accessed: 15-Mar-2020]
46. Reyes F, Cerpa N, Candia-Vejar A, Bardeen M (2011) The optimization of success probability for software projects using genetic algorithms. J Syst Softw 84:775–785, 05