An Approach To Predict Software Application Success Using Voting Ensemble Method

Sangeeta Rathod[1,*], *Prajakta* Ugalmugle[2,**], and *Ruchita* Waghmare[3,***] *Tabassum* Maktum[4,****]

[1]Department of Computer Engineering
[2]Ramrao Adik Institute of Technology
[3]Nerul, Navi Mumbai-400706,
[4]Maharashtra, India.

Abstract. Mobile app distribution platform consisting of Google play store and Apple Store gets covered with several hundreds of new apps every day with many more enthusiastic developers working independently or in a crew to make them successful. With huge competition from all over the world, it is vital for a developer to recognize if he is proceeding in the proper direction or not. It is not like making a film wherein presence of famous celebrities increase the chance of success even earlier than the movie is released, it is not the case with developing apps. Since maximum Play Store apps are free, the revenue version is pretty unknown and unavailable as to how the in-app purchases, in- app advertisements and subscriptions make a contribution to the fulfillment of an app. Thus, a software's success is normally decided through the wide variety of installs and the star ratings that it has acquired over its lifetime in place of the sales it generated. So in order to test if the app is assembly the expectancy of human beings we need a software that will check that apps evolved are successful or not. The framework for predicting success of software application is proposed in this paper. It is a software which will provide the achievement of app/software program relying not only on number of install and star rating but will consider all the factors of an application description and customers reviews. The exploratory data analysis to jump in deeper into the Google Play Store information is performed. The relationships with specific functions inclusive of how the wide variety of phrases in an app call for instance, affect installs are used to use them to find out which apps are much more likely to succeed. Using those extracted functions and the of sentiment of customers the proposed method will predict the "success" of an application using Google Play Store Data. The algorithm applied are Support Vector Machine, K-Nearest Neighbour, Decision Tree and Random Forest. In order to improve accuracy the Voting Ensemble technique is applied in proposed method. The accuracy of various algorithm is compared to judge the performance of proposed method.
Keywords : hyperplane, dataset, Support Vector Machine, Exploratory Data Analysis, voting ensemble

1 Introduction

Software industry is at its peak in this era. It has been discovered that the giant growth of the mobile application marketplace has an outstanding impact on digital technology. With huge competition from all around the world, it is vital for a developer to realize that if he is going in the perfect direction. The Google Play Store is found to be the largest app market in the world. It has been observed that although it generates more than double the downloads than the Apple App Store but makes only half the money compared to the App Store. An application's success is generally determined by the number of downloads and the user ratings that it has received over its lifespan rather than the profit it generates and rest features on which applications success depends. Two app market places which saw a hockey stick curve in their business were the following:

- The Google Play Store
- The Apple App store

These became home to millions of apps which were accessible worldwide. The access to these apps led to increase in the use of apps. More apps were needed for people who were spending increasing amount of time on their smart phones. This gave developers an opportunity to develop android and iOS applications for smart phones. The app market is estimated to cross 77 billion dollars in revenue. Looking at this lucrative opportunity, many developers jumped into app development. But, now the market place is over saturated. There are millions of apps in a single category. What once started as a blue ocean is now turned into a bloody red ocean, which is how things were predicted to be and does not really come as a surprise. There still needs to be some work around for the developers who are enthusiastic about the app market because we never know, the next company grossing the trillion dollar mark might be an app launched in the saturated app market, but which was solving a problem

*e-mail: sararathod000@gmail.com
**e-mail: prajaktaugalmugle7@gmail.com
***e-mail: ruchi.r.c.777@gmail.com
****e-mail: tabmaktum@gmail.com

or providing a service which no one else looked at. So developers still need to have an opportunity to try their hand out at the next idea of the app they have. But the reality is that app development is time-consuming. A developer might spend 4 months developing an application, which when launched on the app store would hardly have 500 users. To avoid this scenario the proposed method is given in this paper for predicting whether the developer is creating an application needed in the market place and will be successful or which will fail in the market. To check the success rate of any software application the data from the store should be collected where the app is being uploaded and made available to user. Depending on a single prediction algorithm will tend to give inaccurate result. The proposed method given in this paper uses four algorithm Support Vector Machine, K-Nearest Neighbour, Decision Tree and Random Forest to predict success rate of an application. In order to approve accuracy the Voting Ensemble Technique is applied in proposed method.

The paper is organized as follows- section 2 is literature survey on the Software Application Success Predictor which summarizes all the existing approaches and papers. Section 3 presents the information about the proposed method used in this paper. The detail methodology/techniques is discussed in this section. Section 4 presents Experiments and Results of the System which gives the details of observations, results and model outcomes. Further conclusion is given this paper.

2 Literature Survey

This section reviews some of the existing techniques for predicting the success rate of any applications. The algorithms and techniques used by diverse papers are elaborated in subsequent section In paper [3], the author scraped information from the Google Play Store to extract as many capabilities as viable the use of an internet crawler 'scrapy' and used the statistics to train three models to predict the success of an application. To predict success metrics of an app, revenue must be the key feature however as it isn't always located publicly, the author used the number of installations and average user score, skilled linear model to categorize whether an app might be a successful or not and moreover used linear regression to predict the average rating of a system. The principal factor evaluation was performed in order to focus on variation and create strong styles of the dataset by the use of inputs of GLM and Linear regression models. Using these models, the author concluded that approximately 35% of total successful applications has the word 'photo' within the description and approximately 31% has the word 'share', the usage of these models, the developer can be cited the genres of application which are primarily liked by end users. For future work, the author suggested using the revenue to predict success metrics and referred agencies like App Annie to accumulate such data. From the defined success metric, a developer may be capable of find the economic achievement of an application. This paper acted as our reference paper, we got the idea of how the wide variety of downloads and average rating can play the crucial role at the same time as defining success metric and to find the success rate of any application.

Further more in paper[4], the authors tried to represent the review-rating mismatch, through establishing multiple systems which can routinely hit upon the inconsistency between these two, to show this mismatch they carried out two machine learning approaches, it included different classifiers like Naive Bayes Classifier, Decision tree, Decision stump, Decision table, and few other algorithms, and the other approach centered on deep gaining knowledge of techniques. They also hosted numerous surveys with the intention to research the opinion of end customers and builders concerning this mismatch. The outcome of the survey was pretty expected, each the developers and end users of android app agreed that rating of an app should match with its corresponding review, and they also asserted to have an automated device to detect the mismatch between rating and review if there is any. This paper proposed a notable way to symbolize the review-rating mismatch, we got inspired by way of their work and gave an effort to create a feature based on the review and rating mismatch and remove the ones which has a better distinction in mismatch but because of time constraint three of our members could manually annotate only 2000 reviews out of 199763 individually, due to this massive variance we couldn't get a proper result and dropped the idea of doing so.

Another paper working at the same aspect [5], the author states that the numeric rating as inside the stars given by users have a large distinction than in comparison to the reviews given by means of them consequently a rating system has been proposed by using the author a good way to dispose of the ambiguity created by means of the mismatch of the score and respective review by same user. It has been visible that how much we as users are dependent on others opinion while taking any decision so in accordance to the author people download the app based on the grading that is given for that particular app. Here the author describes the trouble dividing it into two subproblems. Firstly, the vagueness and secondly it's miles the biasness to the summarized rating of the users. Further explaining the problem, he brought that earlier people used to extract the rating from the remarks instead consisting of the star rating associated with it. To clear up the problems, he proposed a technique that it will initially conduct sentiment analysis on the user reviews and will generate a numeric rating from the polarity. Thus, a final rating could be the average of the rating from the sentiment analysis and the star ratings that are given through the users. This concept would reduce the confusion of the users and permit them to have a final rating primarily based on both review and star rating. This paper shows a strong relationship into the star ratings and the reviews of the users, which helped us of selecting up the idea of using the reviews of the app in our work.

Correspondingly, in paper [6], the author proposed a framework for mobile application developers with which they may be capable of bring in modification on functions

the ones are located negative primarily based on the end user's evaluation in their application. Their framework consisted of three main building blocks (i)subject matter modeling, (ii)sentiment analysis and (iii)summarization interface. This body work changed into designed to make developers comprehend that sentiment rating presents an extra accurate cost of user feedback for an application than star rating and how important it's far to take account of the biasness of the capabilities that affect the overall score of an application. This paper guide us to find the sentiment mean by showing how it delivers some more accurate value than star rating.

Considering the truth that how important polarity values are, as a consequence from paper [7], it helped us to get the concept of removing the inconsistent reviews that will basically reduce noise from the dataset and give better overall performance in sentiment analysis, and consequently will generate polarity value more accurately. They proposed WisCom, a system this is able to analyze at least ten million user ratings and comments inside the app markets in three exceptional levels. The first feature of their system is to become aware of the inconsistencies in reviews; secondly, they have looked for the reasons why people do not like a selected app and the way the opinions change over the time. They prolonged their analysis to provide a valuable perception into the whole app market providing users with fundamental concerns. It generates techniques for summarizing and mining the reviews, which will help the end users to select the exceptional app that has the best experience without analyzing the comment, also app developers can then use the set of end result to apprehend why end users don't like their app which will eventually help the developers to improve their quality of app. More specifically it analyzes reviews on 3 levels which are firstly on single review (micro level) done the usage of regularized linear regression model, secondly on reviews of each app (meso level) by way of doing LDA in addition to they accomplished topic evaluation on distinctive time segments to choose how review modifications over the time, and ultimately on all the apps within the market (macro level). They highlighted all their three-level strategies and described the benefits so as to be obtained by the end-users, developers, and the entire mobile ecosystem.

In paper [8] the authors tried to symbolize the review-rating mismatch, started investigational and exploration journey with pre- processing job of two hundred random terms, taken from Indian agriculture area based guide Hindi dictionary database. Apart from English SVM successfully carried out for identification of relations in Indian Languages from the IndoWorldNet. These classified relations can be used for the generation of established hierarchical version as ontology of particular domain conforming to the idea of semantic web.

This paper [9] proposed the combination proceed towards POS tagging, SVM classifier and FCA-based domain ontology intend to increase the sentiment classification that is training the capabilities based on the supervised learning which categorised the opinions or sentiments within the reviews or comments as positive or negative on every feature. By the use of this method we can view the strength or weak point of the products in more detail. We also looked in papers which can be of different domains however of comparable work to understand better. The main assessment criterion of the overall performance is the predicting accuracy of all users in test set.

Though there are many existing system available in literature for predicting success of software application they lack in terms of accuracy and efficiency, hence an approach to accurately predict success of software application by applying four machine learning algorithm and Voting Ensemble method is proposed in this paper.

Though there are many existing system available in literature for predicting success of software application they lack in terms of accuracy and efficiency, hence an approach to accurately predict success of software application by applying four machine learning algorithm and Voting Ensemble method is proposed in this paper.

3 Methodology

The overall process of proposed system is given in figure 1. The functionality of all module is discussed below.

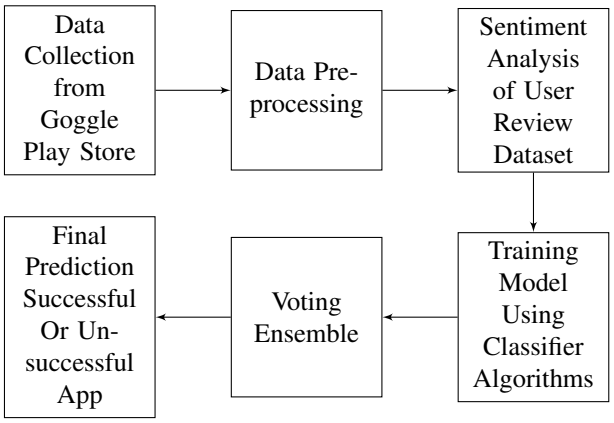


Figure 1. Block Diagram of the system

3.1 Data Collection

First of all, to gather knowledge of application with all its facts we have used Google Play Store platform. Google Play Store data can be acquired from Kaggle dataset or Web-scraping. Scraping is a technique of statistics mining from websites that uses software package to extract all the facts accessible from the targeted web site through simulating human behavior. This method largely focuses on the transformation of unstructured information (HTML format) on the websites into structured information (database or spreadsheet). The Beautiful Soup 4 (BS4) is used to extract knowledge from Play Store. Beautiful Soup 4 is a Python library for pulling knowledge out of hypertext markup language and XML files.

3.2 Data Preprocessing

All the app details have been scraped, there were no null or missing values in the dataset. Moreover, the raw in-

Table 1. features of an app

FEATURES	DESCRIPTION
name_app	This is the name of an application
cat_app	It represents the category of an app
n_review	The total number of reviews for the app given by user
rating_app	It is star rating given by user for the app
size_app	Size of an app in mb or kb
n_installs	It represents the total no of downloads for the app
type_app	It indicates whether the app is free of paid
price_app	It represents the cost for usage of the app
user_group	It represents the age group of users who are using app
genre_app	It represents genre of the app
last_updated	It is last date of updating
current_ver	It represents the current version
android_version	It is android version for the app

formation is extracted manually so there is a need of preprocessing to turn it into some utilizable information. So it is able to perform EDA and run algorithms on dataset installs, total number of ratings, rating distribution were also converted to integers. Sizes of apps having kilobyte were converted to megabytes to have consistent units. Apps with varying sizes (those that vary with device) were set to the average size of the rest of the apps. Later the categorical attributes like category or genre, content rating and type were label encoded, for instance free apps were labeled as 1 and paid apps were labeled as 0.

3.3 Feature Extraction

Feature Extraction aims to take out knowledge from the database by creating significant features from the existing ones. These new set of significant features should be able to sum up the information contained in the initial dataset. The table 1 shows the list of features in dataset.

3.4 Sentimental Analysis

Sentiment analysis is the measurement of beneficial, non beneficial and neutral language. It can evaluate how a person feels by their emotions, opinions and attitude. It can also be referred to as opinion mining. It basically inspects the issue of reading texts, like posts and thoughts uploaded by users on diverse platforms, meetings and electronic businesses regarding the 15 opinions they have about a product, service, event, man or woman or their thought by way of taking out the individual information from the source information.

3.5 Classification and Prediction Algorithm

3.5.1 K nearest neighbour Algorithm

The K nearest neighbour Algorithm is one of the frequently used classification algorithm. This algorithm divides the entire dataset into different classes and then class of any new sample can be predicted by their neighbour. It can also be used for regression output which has output in continuous domain. This value chosen is the average of the values its nearest neighbors.

3.5.2 Random Forest Algorithm

The Random forest is a ensemble model consisting of many decision trees. Random forest algorithm is prone to overfitting. Random Forest algorithm gives more accurate result than decision tree algorithm that is the reason it is widely used. It ensembles many random trees to give high accuracy. Each tree in the forest predicts the category to which the new record belongs. Finally, the new record is given to the category that wins the majority vote this is the basic task performed by random forest algorithm.

3.5.3 Support Vector Machine

The Support Vector Machine is a supervised and linear Machine Learning algorithm mostly referred as SVM. We have used SVM to train our model for predicting success of an application as SVM gives more more accuracy as compared to other algorithms such as logistic regression, Decision Tree. In SVM we have used SVC - Support Vector Classifiers which classifies apps as successful or unsuccessful by finding optimal hyperplane.

3.5.4 Decision tree

The Decision tree is a famous algorithm used for regression and classification. The model for classification is prepared in the form of tree like structure. The process of building decision tree is incremental. The leaves in the tree represents the decision for the given sample. The intermediate node contains the test for different attributes. There can be 2 or more than two branches based on the type of attributes. It accepts both numerical and categorical attributes.

3.6 Voting Ensemble

Voting Ensemble is a meta-classifiers for combining weak machine learning classifiers into one robust, accurate classifier for classification and prediction. Ensemble learning is mainly used to improve the performance of the model, or to lessen the selection of a poor one. Training a model with one algorithm will may not predict the outcome as accurate as combining two or three classifier into one meta-classifier and then training model to predict the outcome. Voting Ensemble combines different classifiers outputs and takes vote among them to predict the accurate output. Voting Ensemble prediction method is more powerful as compare to other prediction algorithm which improves precision, predictive power, accuracy and lessens error rate.

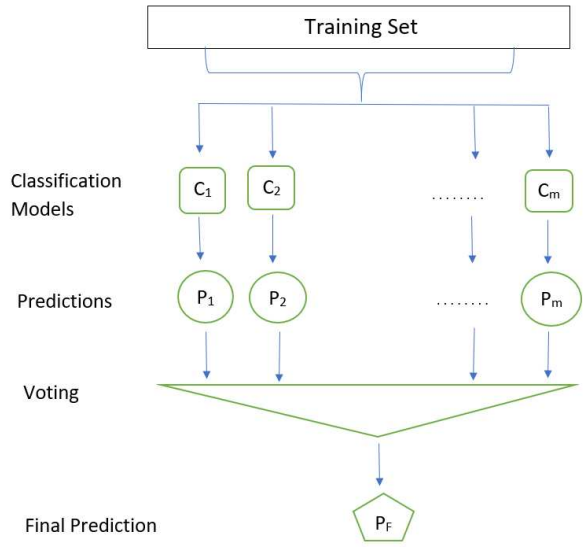


Figure 2. Voting Ensemble

3.6.1 Hard Voting Ensemble

In Hard Voting Ensemble the class which gets the highest voting is chosen. Consider that, there are n classifiers such as C1,.....,Cn. The output class label l is predicted for any test data by considering the majority voting of classifiers Ci where i = 1,...,n.

$$l = mode\{C1(x), C2(x), ..., Cn(x)\} \quad (1)$$

Example : Let there be 3 classifiers the outcome given by each classifier is given below:

- C1 = 1
- C2 = 1
- C3 = 0

The output of the voting ensemble would be:

$$l = mode\{1, 1, 0\} = 1 \quad (2)$$

3.6.2 Soft Voting Ensemble

The another method for finding output class label for test data is soft voting in which rather than giving outcomes as 1 or 0, each classifiers gives probability for class 1 and 0. The output class label l for any test data is predicted by taking the average of the probabilities given by classifiers Ci where i = 1,...,n.

Example : Let there be 3 classifiers, the probabilities given by each classifier for class 1 and 0 is given below:

- classifier = [class 0, class 1]
- classifier C1 = 0.2, 0.8
- classifier C2 = 0.4, 0.6
- classifier C3 = 0.6, 0.4

Class 0 = (0.2+0.4+0.6) / 3 = 0.4

Class 1 = (0.8+0.6+0.4) / 3 = 0.6

Therefore the output of soft voting for given test data will be 1 at threshold 0.5.

Steps to develop Voting Classifier for Voting Ensemble

1. Import required classifiers and module from respective library.
2. Import VotingClassifier from sklearn.ensemble library.
3. Load the desired dataset.
4. Create classifiers Ci where i = 1,...,n.
5. create voting classifier for Ci
   voting_classifier = {
   (labels(Xn),
   (C1,C2,...Cn),
   voting=hard OR soft }
6. train and test voting classifier.

4 Experiments and Result

Dataset Description :
The proposed system is trained and tested over the dataset taken from Google Play Store. There is total 10842 application data in our dataset which is obtained from Google Play Store. After Data cleaning and pre-processing dataset was shrunk to 8892 application because of some incomplete data and null values. Among 8892 apps 4568 applications are labelled as successful after running the success matrix function and 4324 apps are labelled as unsuccessful applications.

In order to do the sentiment analysis on google play application user reviews dataset analysis was performed with two algorithm to see which gives the more accurate result.

The SVM and naive bayes algorithm are used to perform sentiment analysis. The table 3 and 4 shows the confusion matrix for both algorithm with respect to their accuracy.

	precision	recall	F1-score	support
0	0.99	0.99	0.99	12500
1	0.99	0.99	0.99	12500
micro avg	0.99	0.99	0.99	25000
macro avg	0.99	0.99	0.99	25000
weighted avg	0.99	0.99	0.99	25000
accuracy:	0.99204			

Figure 3. Confusion matrix for sentiment analysis using SVM algorithm with accuracy 99%

	precision	recall	F1-score	support
0	0.79	0.88	0.83	12500
1	0.87	0.77	0.81	12500
micro avg	0.82	0.82	0.82	25000
macro avg	0.83	0.82	0.82	25000
weighted avg	0.83	0.82	0.82	25000
accuracy:	0.825			

Figure 4. Confusion matrix of naive bayes algorithm with accuracy 82.5%

For 80/20 testing and training dataset a `train_test_split` function is used. So 7116 application are reserved for training and 1776 apps for testing purpose. Among 7116 training data 3558 apps are successful and 3558 apps are unsuccessful, and among 1776 apps 1010 apps are successful and 766 apps are unsuccessful apps. This is done in order to avoid overfitting and underfitting.

For success prediction the model is trained with four algorithms - SVM, Random Forest, K-NN algorithm, Decision Tree.

	precision	recall	F1-score	support
0	0.60	0.77	0.67	766
1	0.78	0.61	0.68	1010
macro avg	0.69	0.69	0.68	1776
weighted avg	0.70	0.68	0.68	1776
accuracy:	0.678			
AUC-ROC:	0.721			

Figure 5. Confusion matrix of SVM prediction algorithm

	precision	recall	F1-score	support
0	0.76	0.95	0.84	766
1	0.95	0.77	0.85	1010
macro avg	0.86	0.86	0.85	1776
weighted avg	0.87	0.85	0.85	1776
accuracy:	0.848			
AUC-ROC:	0.946			

Figure 6. Confusion matrix of KNN prediction algorithm

	precision	recall	F1-score	support
0	0.68	0.70	0.69	766
1	0.77	0.74	0.76	1010
macro avg	0.72	0.72	0.72	1776
weighted avg	0.73	0.73	0.73	1776
accuracy:	0.725			
AUC-ROC:	0.796			

Figure 7. Confusion matrix of Random Forest prediction algorithm

	precision	recall	F1-score	support
0	0.65	0.76	0.70	766
1	0.79	0.69	0.74	1010
macro avg	0.72	0.73	0.72	1776
weighted avg	0.73	0.73	0.72	1776
accuracy:	0.723			
AUC-ROC:	0.803			

Figure 8. Confusion matrix of Decision Tree prediction algorithm

According to table 5, 6, 7 and 8 it can be observed that the single algorithm can not predict the success of an application on his own with great accuracy. Therefor Voting Ensemble Method is proposed which is a meta classifier which will predict the final result by the method soft voting ensemble by analyzing these four algorithms.

	precision	recall	F1-score	support
0	0.93	0.97	0.95	766
1	0.97	0.95	0.96	1010
macro avg	0.95	0.96	0.95	1776
weighted avg	0.96	0.95	0.96	1776
accuracy:	0.955			
AUC-ROC:	0.991			

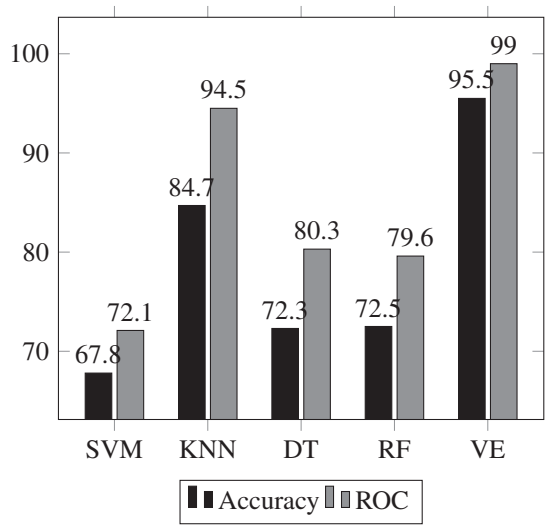
Figure 9. confusion matrix for Voting Ensemble prediction with accuracy of 95%

Figure 9 is the confusion matrix of voting ensemble. It can be observe that voting ensemble is the best method of predicting success of an app with as it gives 95% accuracy.

The Graph 1 shows the graph of performance comparison of SVM, KNN, Decision Tree, Random Forest with voting ensemble.

```
SVM - Support Vector Machine
KNN - K-nearest neighbor
DT - Decision Tree
RF - Random Forest
VE - Voting Ensemble
```

Graph 1. Algorithm Comparison



The accuracy of proposed model is high as compared to other models. Thus the proposed model outperforms the other algorithm. The voting ensemble method helps to correctly predict the success of any software application.

5 Conclusion

The Google Play Store is the most important app market in the world. It generates more greater number of downloads than the Apple App Store, however makes simplest half of the revenue compare to the Apple App Store. So the statistic data from the Play Store is scraped to conduct studies on it. There are a couple of troubles with the Play Store data that have been identified from exploratory data analysis. An inclusion of a function that lets users notify developers if they're unsatisfied with a new update, if they may want to go along into not plummeting an app with low installs to the ground. There is also the trouble of rating mismatch on a smaller scale. If this problem can be mitigated, the Play Store would provide an extra accurate representation of user sentiment which in turn should help developers make adjustments and update to their app accordingly. The proposed model in this paper uses Decision Tree, Support Vector Machine, k-Nearest Neighbour and Random Forest algorithm to predict the success of an application. Finally by assembling these four algorithms by Voting Ensemble Method the model predicts success rate of software application with 95% accuracy which is higher as compared to all four algorithms.

References

[1] *https://www.business2community.com/mobile-apps/2017-mobile-app-market-statistics-trends-analysis-01750346*, (2017)
[2] *https://entrepreneurscan.com/blog/help-my-ocean-is-turning-red/*, (2018)
[3] Tuckerman, C., (2014).
[4] Aralikatte, R., Sridhara, G., Gantayat, N., and Mani, S., (2018).
[5] Islam, M. R., (2014).
[6] Luiz, W., Viegas, F., Alencar, R., Mourão, F., Salles, T., Carvalho, D., Gonçalves, M. A., and Rocha, L., (2018).
[7] Fu, B., Lin, J., Li, L., Faloutsos, C., Hong, J., and Sadeh, N., (2013).
[8] Megha Garg,Bhaskar Sinha,Somnath Chandra, (2015).
[9] Khin Phyu Phyu Shein ,Thi Thi Soe Nyunt, (2010).
[10] Ziheng Wang, Yonggang Qi, Jun Liu, Zhanyu Ma, (2016).
[11] Chih-Chung Chang and Chih-Jen Lin, (2019).
[12] *https://www.kaggle.com/lava18/all-that-you-need-to-know-about-the-android-market*.
[13] *https://www.vaishalilambe.com/blog/data-science-algorithms-random-forest*
[14] *https://slideplayer.com/slide/7364467/*
[15] *https://stackabuse.com/implementing-svm-and-kernel-svm-with-pythons-scikit-learn/*