Letters in High Energy Physics
ISSN: 2632-2714 Volume 2024

Predicting Mobile App Success Using a Robust Hard Voting Ensemble Learning Approach

Aqsa Saleem[1], Muhammad Aleem[2*], Nayem Uddin Prince[3], Md Mehedi Hassan Melon[4], Salman Mohammad Abdullah[5], Shah Md. Wasif Faisal[6], Mohd Abdullah Al Mamun[7]

1) Department Of Computer Science, COMSATS University Islamabad - Sahiwal Pakistan
2) Faculty Of Computing Universiti Malaysia Pahang Al-Sultan Abdullah 26600 Pekan, Pahang Malaysia
3) Daffodil International University, Bangladesh
4) International American University, Los Angeles
5) Washington University Of Science and Technology, USA
6) Washington University Of Science And Technology, USA
7) MBA In Information Technology Management, Westcliff University, USA

Email: aleemmian380@gmail.com

Abstract. Apps have become an inseparable part of our daily lives. There are different kinds of mobile apps on the market. Google Play Store apps development is one of the most enticing and consumer-friendly development paradigms for mobile apps. On the other hand, the paradigm is still in its early phases and does not address critical issues such as an app's success and failure. A considerable number of mobile apps do not acquire a good solution, squandering stakeholders' time and effort. Therefore, predicting the success of a new app will be helpful for developers. This research proposes an ensemble learning-based approach for predicting the success of the mobile app. For this purpose, the app's important attributes (rating, number of installs) can be selected from the dataset. Datasets can be preprocessed using NLP (Natural Language Processing) technologies and perform Data Analysis. These selected features were then deployed to several ML (Machine Learning) algorithms — Deci- Sion Tree Classifier, Random Forest Classifier, K - Nearest Neighbor Classification, Gradient Boosting Classifier, and Light Gradient Boosting Classifier. Finally, an ensemble model pro- poses to predict a new app's success. Our suggested model outperforms, with an accuracy of96.772239%.

Key-words: Ensemble learning, Natural Language Processing (NLP), Mobile Apps, Data Analysis, Decision Tree, Random Forest, K-Nearest Neighbor, Gradient Boosting, Light Gradient Boosting

1. Introduction

Mobile applications are used more rapidly and conveniently because of the widespread use of smartphones. As a result, mobile apps have grown to be extremely vast and competitive. Al- most all digital requirements have spawned multiple apps, giving smartphone users a plethora of options when it comes to which app to download and use. Many characteristics of an application could be crucial in deciding the popularity and success of an application in such a competitive environment. As a result, mobile app developers must consider various factors when develop- Ing and deploying their apps. Although the app store has millions of apps, most of them are rarely downloaded or used. So it is very important to investigate how apps are becoming a vital part of people's lives. The rapid growth of the mobile app market has a significant effect on the development of advanced innovation. However, as the flexible application market continues to grow, so does the number of portable application designers, resulting in the global portable application sector earning as much as it can. Mobile applications have become the key usage for smartphones, providing a variety of features and services like social networking, entertainment, purchasing, checking information, and navigation [1].

According to anecdotal evidence, users install an average of 63 applications on their cell- phones and spend more than 6 hours each day on them [2]. IOS systems have been launched
1.85 million applications worldwide due to the popularity of mobile applications; the number of applications available for Android systems is slightly higher, with approximately 2.56 mil- lion applications available as of the first quarter of 2020 [3]. Among the numerous applications accessible, it is worth mentioning that only the top applications in each category comprise the bulk of the user's time. In another way, the overwhelming number of mobile applications fails to meet expectations.

Furthermore, nearly half of smartphone users only use 5–10 applications each day and uninstall 3.9 applications every month. In this circumstance, app developers and service providers must figure out how to attract and retain consumers to succeed in the mobile application market.

An app store, like Google Play, is a platform where users may download mobile applications for every service or software. Users can explore and download apps via app stores, while de- elopers can maintain track of their apps by providing reviews and star ratings. Reviews may contain the user's experience, problem reports, requests for additional features, or a numerical rating of the app [4]. Various app features, both internal and external to the app, can play a key influence in assessing whether or not the app will succeed. Using a dataset comprising in-formation gathered from the Google Play Store, this research investigates several characteristics external to an application to classify the application's success. Several classification models are developed and analyzed in order to gain a better knowledge of how to classify the success of an app [5]. The author of another paper [6] examines the application ratings. It highlights the fact that the way a store is rated after reaching a particular limit has no effect on the overall store rat-Ing, even after users have rated it. It emphasizes that the updated app rating depends on the app's version. As a result, the notion of grading variations arose. The available store rating can be used to calculate version ratings. Google Play Store is becoming one of the most appealing and user-friendly platforms for Mobile apps among all mobile app distribution platforms [7].

A large number of new Mobile apps are added to this platform on a daily basis. Given the tremendous challenges around the world, a designer must realize that they are on the right track [8]. To maintain this revenue and marketplace, application developers may need to know how to maintain their current position. The Google Play Store is considered the most extensive application platform. However, the Google play store paradigm is currently in its early stages and has some obstacles that must be overcome. One of the most significant issues it encounters is that it directly impacts the app's success rate [9]. As a result, many apps are either abandoned or fail to yield satisfactory results. Developers are constantly seeking to improve the effectiveness of their new apps through solo or teamwork. Developers find it challenging to maintain their market position on this platform as it becomes more competitive. However, if developers can predict the success of a new app before it is released on the Google Play Store, they may find it easier to compete in this more competitive market. Despite the massive expansion of apps in recent years, there is a scarcity of related work on app analysis. In this paper, we investigate the extent to which we can predict an app's success based on its different attributes like rating, reviews, app size, and price. We make extensive use of ensemble learning techniques for this purpose.

The generic meta-approach to machine learning known as" ensemble learning" aims to achieve optimum prediction performance by integrating a variety of approaches to get the best accu-racy [10]. Individually, various ML algorithms may not be able to produce optimum results; consequently, integrating them will enhance the model's performance and improve its accuracy. The Ensemble learning approach for predicting and classifying the success of the mobile app has been shown to perform better than using a single classifier. Bagging, stacking, and boosting are the three classes of ensemble learning. Bagging involves making several decisions on different samples of the same dataset and calculating the mean prediction, [11] while stacking involves fitting different models to the same data and learning the aggregated predictions using another model [12]. Boosting involves adding ensemble members successively to correct previous pre- dictions provided by other models. Then the average of the predictions is taken [13].

To this end, we explore the app descriptions in-depth and present a unique technique based on ensemble learning for predicting the success of mobile apps. The suggested method classifies ap- plications as either successful or unsuccessful. First, we obtained a dataset from Kaggle and used NLP technologies for preprocessing. Second, we perform Data Analysis. Third, multiple ML classifiers are used for the successful prediction of the app. Finally, the voting ensemble-based technique is proposed to produce a final prediction. The core objective of this voting method is to reduce algorithmic errors by combining the decisions using a majority voting technique. Four separate performance metrics: Precision, Recall, Accuracy, and F1-Score, are used to provide an effective and comprehensive validation of the performance of models.

Following are the key contributions of this research:

- An ensemble learning technique is suggested to determine how successful a new app will be.

- Machine learning algorithms, including Decision Tree Classifier, Random Forest Classi- fier, K - Nearest Neighbor Classification, Gradient Boosting Classifier, and Light Gradi-ent Boosting Classifier, are used to predict an app's success.
- A Hard voting ensemble learning model is proposed to improve classification performance.
- The precision, recall, and f-measure of the proposed approach are up to 0.985657 0.966535 and 0.976000 respectively. This suggests that the proposed approach is accurate.

2. Literature Review

Magar et al. utilized a Google Play Store-extracted dataset [5]. Various external features were used for an app to classify its success. This study tackles a classification problem. SVM, K-Nearest Neighbor, Stochastic Gradient Descent, and Random Forest were some of the models that were employed in the analysis of data [14]. Based on execution times and performance metrics, the models were compared. This research determined the classification model with the best performance and analyzed the relationship between target categories and classification performance. According to the literature, the sales price, number of updates, languages in which the app is released, operating system versions supported, API functionalities, and the package size are all measurable app features that might positively affect its success [15]. Furthermore, the popularity of the app's category [16] and the user rating tend to have a substantial impact onsuccess.

N. Picoto et al. used multivariate logistic regression [17] to examine the app's performance with five parameters, including user ratings, category attractiveness, diversity, capacity factor, and theatrical release. It also used a fuzzy set qualitative comparative analysis (fsQCA) to look for new causal explanations for the performance of the mobile app. According to multivariate analysis [18], the attractiveness of the subcategory, variety, capacity factor, and app release date are all characteristics that boost the likelihood of an app being ranked among the top 50.

Singh et al. worked using an Android app dataset [19]. They used exploratory data analysis and multiple ML models to figure out which aspects of an app impact its success. Decision Tree, Random Forest, SVM, XGBoost, and KNN were some of the models employed [20]. They attained accuracy rates of 70.49 percent, 80.34 percent, 75.59 percent, 79.99 percent, and 77.26 percent, correspondingly. They determined that an app's rating and content rating were important in determining its performance in the competitive online marketplace. Mueez et al. worked on the Play Store apps dataset to predict an app's ”success” [20]. They defined success in terms of the function of an application's rating and the number of times it has been installed. Another noteworthy feature of their endeavor was the fact that they took into consideration actual user reviews. Data from app reviews were used to make predictions based on a list of commonly used terms in the reviews. They employed those terms in the app to do sentimental analysis [21], and the sentiment score served as a featured column in their predictions. Although the authors made some assumptions when selecting these feature columns, they attained an accuracy of 85.09 % utilizing the XGBoost Classifier, K-Nearest Neighbor, Random Forest, and SVM. The authors' handling of a binary classification issue must have contributed to the excellent accuracy rate obtained.

Dehkordi et al. showed a dataset of 100 successful and 100 unsuccessful Android applications that were available in the Google Play Store [22]. Each app had 34 features. The repos-itory was then used to predict the accuracy of various neural networks and other classification techniques in two scenarios: with and without using the principal component analysis (PCA) algorithm. Without PCA, the NPR algorithm achieved a prediction accuracy of 95.5%, while theMLP method achieved a prediction accuracy of 99.995%. This means that the NPR algorithm ismore accurate when there are more repository features. Suleman et al. used Google Play Store app data and how they were received by the users [23]. The dataset contains 10839 entries, which perhaps an insufficient number is given the problem they were attempting to solve. This dataset is used to train various ML models [24] to predict the app's success. The goal of their study was to figure out which model worked best for predicting app ratings.

Bashir et al. evaluated a dataset of mobile applications available through Google Play to de- termine the level of success each app had in the ”real world” [7]. The researchers concentrated on predicting an application's success before it was released. The number of installs and ap- plication ratings were determined ”success.” Due to the authors' concentration on the ”success “of the apps, they encountered binary categorization problems regarding predictions. When they tried different machine learning methods, they determined that the SVM model made the most accurate predictions

[24]. In recent years, as mobile devices' computational capability has in- creased, the importance of these devices has moved to the software and data they can provide. The expansion of mobile applications is an opportunity and a significant challenge for companies and developers, since it is challenging to stand out among the 7.1 million apps [25] accessible on app stores. Acquiring awareness is a difficult and time-consuming task. The mobile market is difficult to understand due to a lack of data on app demand and sales. Furthermore, because mobile users spend more time using applications than websites, businesses seeking competitive advantage must understand the elements that influence app performance to design appropriate mobile strategies, and effective mobile marketing [15].

Aralikatte et al. attempted to represent the review-rating discrepancy by developing differ- ent algorithms capable of automatically detecting the discrepancy between the two [26]. They employed two machine learning strategies to demonstrate this mismatch: The first approach re- lied on many classifiers and algorithms, such as the Decision tree, Naive Bayes, Decision stump, and Decision table; the second approach relied on deep learning. They also conducted numerous surveys to understand end users' and developers' perspectives on the mismatch. The survey's findings were quite predictable: both Android app developers and end-users agreed that an app's rating should correspond to its related review, and they claimed to have an automatic system in place to detect any discrepancy between rating and review [27].

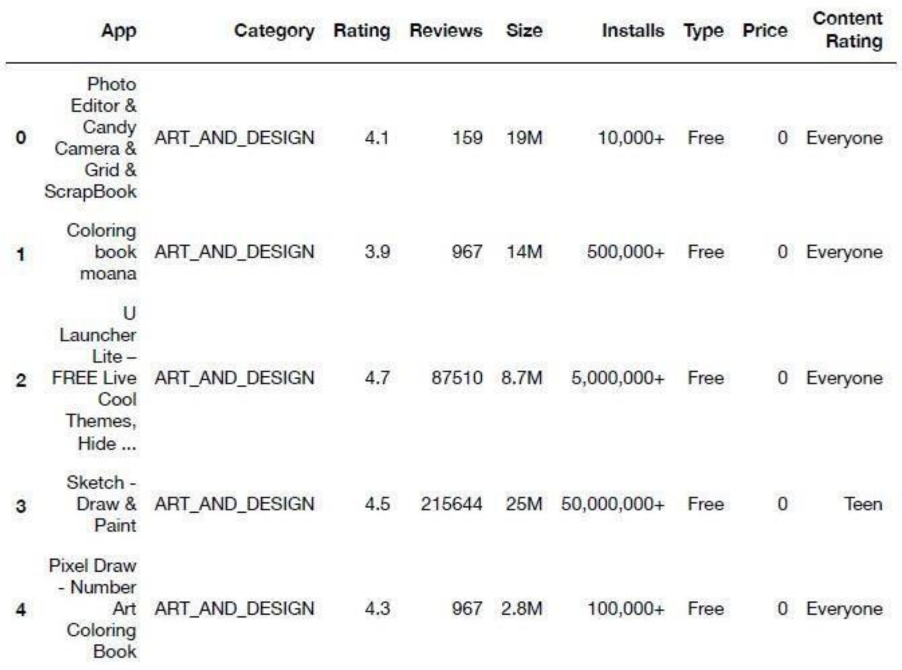
Y. Yao et al. proposed a novel technique for developing an app-recommendation system that considers a vital feature of each app [28]. In other words, they wanted to create an app-recommendation system that was version-aware. As the authors illustrated, app ratings can sub-statically change based on the application's version. This is an essential feature of their approach. The authors built their model and generated recommendations based on the review text for each app. They suggested a single Version-Aware Matrix Factorization (VAMF) framework that con- side red both version-level and overall app-level review text. Researchers apply a greedy method in the article [29]to extract mobile-app elements from official pages displaying programmed de- scrimptions to examine business and technical aspects of mobile applications. Researchers used the hidden model to classify spam in mobile app stores and divided reviews into malicious and non-malicious categories.

B. Fu et al. address the question of why a user does not want an app and provide an expla-nation for why it fails [30]. Not all reviews may be treated and incorporated as part of the study. Such reviews generate a lot of noise and increase the percentage of inaccuracies. Such reviews were attempted to be eliminated by the authors, which reduced noise in the dataset and improved sentiment analysis performance. As a result, a more precise polarity value is generated. Li [30] suggested WisCom, a system capable of analyzing minimum ten million user comments and ratings across three levels in app stores. This technology is believed to have initially identified inconsistencies in reviews before looking for reasons why people dislike a specific application and how reviews change with time in response to customer demands. Their suggested strategy and valuable research present an enthralling tour of the most relevant issues. It generates ways for summarizing and mining evaluations, assisting end customers in selecting the best program without having to read through descriptive reviews of previous users.

R. Mihalcea et al. introduced a Text Rank graph-based ranking mechanism for graphs col-lected from natural language texts [31]. Researchers explore Text Rank's application to two language processing tasks: unsupervised keyword and sentence extraction. The results produced with Text Rank are comparable to those obtained with state-of-the-art frameworks in these do- mains. B. Liu works on feature reviews [32]. He mentions the ineffective use of user ratings and reviews. In addition, the researcher develops several systems that detect the contradiction between the two features mentioned above. The system primarily employs the Naive Bayes Classifier, Decision Tree, Decision Stump, Decision Table, and a few more Machine Learning Algorithms and Deep Learning Approaches. They conducted multiple polls as part of the process to learn what users and developers thought about the two features mentioned above. The idea of matching app review and rating relevancy was agreed upon by both end-users and Developers, concluding with the likes of the author. Following up on previous research on the same sentiment analysis. This work [4] demonstrates that on the app store, there is a distinction between starred user ratings and reviews. As a result, the author recommends a new scoring system that could eliminate uncertainty and distinction between reviews and ratings that the user has produced. According to the author, the user is interested in downloading an app based on their rating. Zhuet al. developed a PHMM app popularity prediction model based on market ranking, user rat- Ing, and user review to measure app popularity [33]. Wang et al. investigated the causes of app competition and proposed EHCM and an evolutionary competition model [34] for predicting app download volumes. Finkelstein et al. explored app popularity by ranking downloads to determine the relationship between prices, ratings, and app popularity.

3. Data Preparation

The dataset used in this study comprises information on Google Play Store apps. Dataset is available publicly on Kaggle. The dataset contains many features, like App, Category, Price, Installs, Size of the app, reviews, Content Rating, Last Updated date, and so on, as shown in 1.



	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Everyone
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyone
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Everyone
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free	0	Teen
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0	Everyone

Fig. 1. Sample Dataset

3.1. Data Analysis

The first and most important step in any prediction is Data Analysis (DA). DA is the pro-cess of using summary statistics and graphical representations to analyze to identify patterns, and anomalies, test theories, and verify assumptions. DA helps us choose the best prediction algo-rithms and models for the data. Before we work on our data, eliminate any null data that might cause noise in prediction. We analyze our findings and identify that the Google Play store offers over 33 distinct types of apps. The category of Tool has the highest number of counts. We elimi-nate the categories with a lower count because they will not significantly impact our prediction or accuracy but will create noise and inconsistencies. Next, several graph functions are used to an-alyze the frequency of average ratings between 0 and 5 and the categories with the most installed.

3.1.1. Category vs. Apps

The Category column in our dataset contains 34 unique categories. 2 illustrates the Bar Chartof various kinds of apps against the categories.

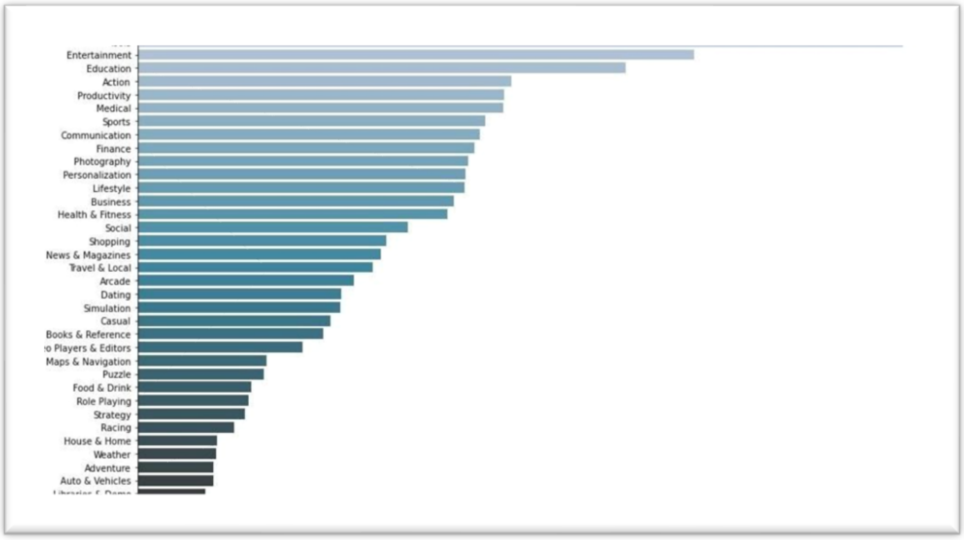


Fig. 2. Bar chart of category against number of app

3.1.2. Category vs. Rating

All the categories have close rating averages. There are many categories of apps that are equal in terms of being the highest rated. The interest should lie within the app categories with the lowest ratings. These poorly rated apps deserve more attention because if a new app in that category were to be put on the app store, the developers could satisfy the demand for innovation in this area. In this case, The Categories of Importance are" AUTO- AND- VE- HICLES" and " ENTERTAINMENT." 3 shows the Bar chart of categories by Ratings.

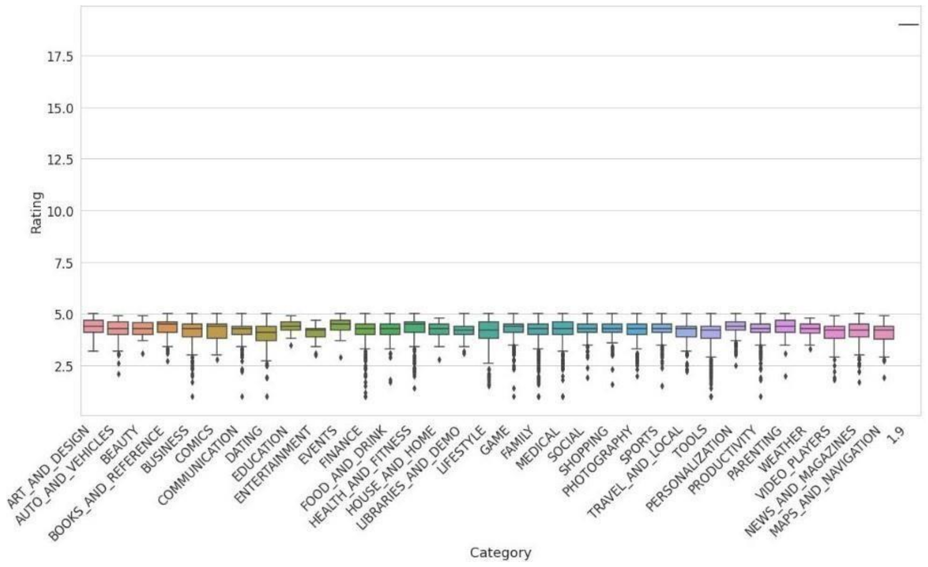


Fig. 3. Bar chart of categories by Ratings

3.2. Data Preprocessing

”Data preprocessing” is an essential part of ”data mining.” The quality of the data has an impact on the outcome. Different mining techniques, such as classification and clustering, give the best and most effective results when preprocessing adequately cleans the data. Most data contain trash values, which should be handled before they impact the performance of predictive models that have been trained. Since we collected data from Kaggle, which is semi-structured or unstructured data that includes large redundant information, there were needed to be prepro-cessed to turn it into some valuable information. NLP Techniques are used to preprocess the data. Tokenization, stop word removal, and lemmatization are some of the preprocessing approaches used to preprocess our data.

3.2.1. Tokenization

: Splitting text into tokens (words) is referred to as tokenization. We break down the text into words and eliminate special characters such as punctuation marks.

3.2.2. Stop Words Removal

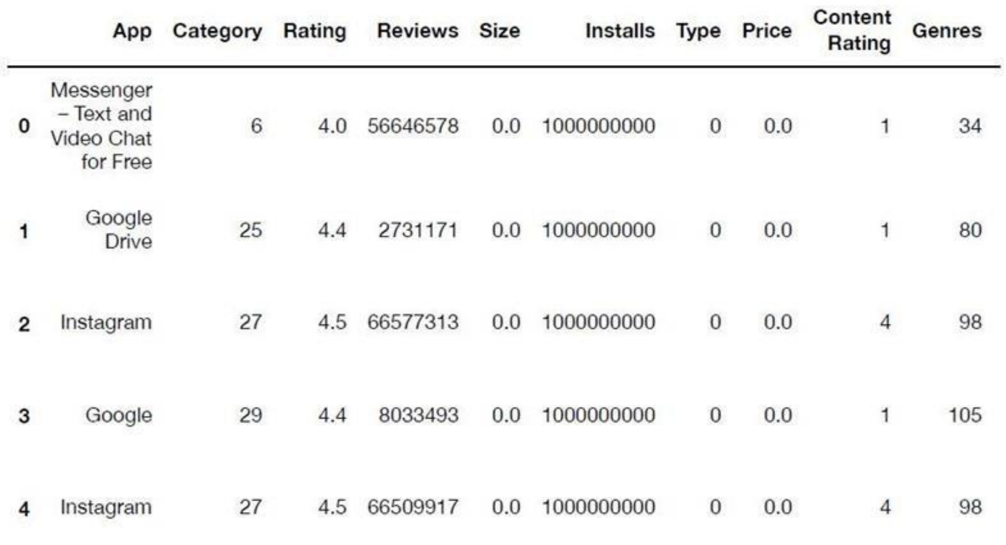
: Words that are employed to make sentences meaningful but have no significance on their own are frequently found in textual documents. These words are known as ”stop words.” During tokenization, we eliminate such terms from the extracted words.

3.2.3. Lemmatization

: Lemmatization is the process of converting superlative and comparative words into their basic words. For example, lemmatization transforms the word” likes” into ”like.”

3.3. Dataset After Preprocessing

As we see above in figure 1, the dataset collected from Kaggle is in raw form. Therefore, EDA and preprocessing techniques were used on the dataset to turn inconsistent data into a com- prehensible form. After performing EDA and preprocessing techniques on our dataset, Installs and content ratings were changed to integers. The app’s size is given as a ”string.” It must be con- verted into a numerical value. To maintain consistency, the app’s sizes in kilobytes were changed to megabytes. The price information is provided as a string. To convert data into the numeric format, the dollar sign must be removed from the text. Dataset after preprocessing is show in 4.

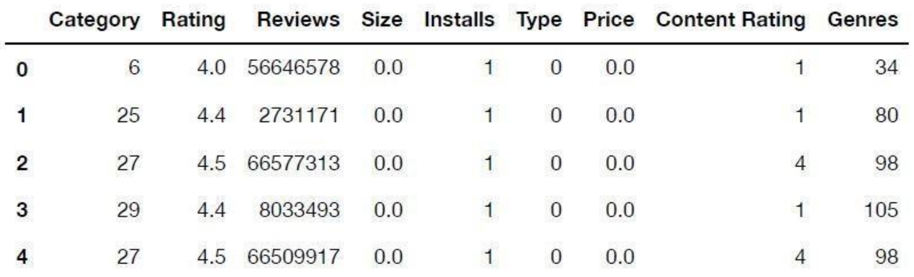


	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres
0	Messenger – Text and Video Chat for Free	6	4.0	56646578	0.0	1000000000	0	0.0	1	34
1	Google Drive	25	4.4	2731171	0.0	1000000000	0	0.0	1	80
2	Instagram	27	4.5	66577313	0.0	1000000000	0	0.0	4	98
3	Google	29	4.4	8033493	0.0	1000000000	0	0.0	1	105
4	Instagram	27	4.5	66509917	0.0	1000000000	0	0.0	4	98

Fig. 4. Dataset After Preprocessing

3.4. Label Encoding

Categorical features including content rating, genre and type are label encoded. The way in which we go about preprocessing the data is by binarizing the Installs column. Anything above 100,000 will be considered equal to 1, and everything below that threshold will be equal to 0. 5illustrates the dataset with label encoding.



	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres
0	6	4.0	56646578	0.0	1	0	0.0	1	34
1	25	4.4	2731171	0.0	1	0	0.0	1	80
2	27	4.5	66577313	0.0	1	0	0.0	4	98
3	29	4.4	8033493	0.0	1	0	0.0	1	105
4	27	4.5	66509917	0.0	1	0	0.0	4	98

Fig. 5. Dataset with Label Encoding

3.5. Feature Selection

Feature selection is choosing a feature subset based on certain criteria. It is also known as attribute selection or variable selection. Feature selection process is show 6. This process makes a dataset more useful and efficient by getting rid of features that are useless, duplicated, or cause noise. The segmented dataset is run through feature selection algorithms to find the most im- portent properties that affect an app's success. We select important features from the data that give us an optimal prediction. After that, we generate a dataset using those chosen features. We split our data into dependent Y and independent X variables. No. of installs column is used as target feature, which consists of quantitative data. After feature selection, the ensemble learning classifier is used to train the data and get high accuracy.

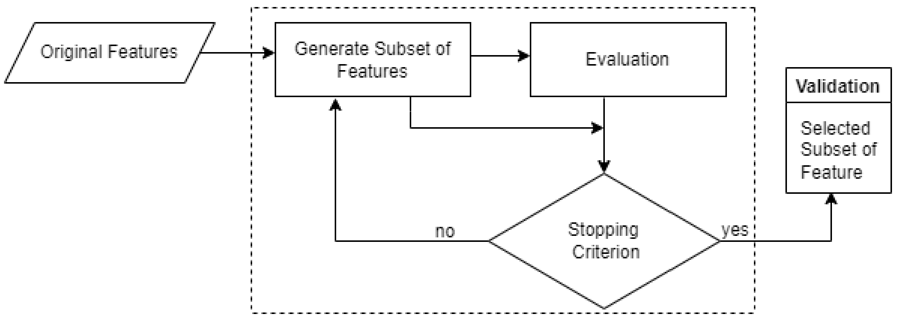


Fig. 6. Feature Selection Process

4. Approach

Overview

The Google Play Store applications dataset is used to predict the success of an app. We propose an ensemble learning model for selecting the dataset's most discriminating features. After selecting the attributes, we preprocessed them and applied multiple ML algorithms, in- clouding Decision Tree Classifier (DTC), Random Forest Classifier (RF), K – Nearest Neighbor Classification (KNN), Gradient Boosting Classifier (GBC), and Light Gradient Boosting Classi- fire (LGBC). Additionally, a voting ensemble-based approach to making a final prediction has been considered. The primary objective of this voting strategy is to eliminate algorithmic errors by combining decisions through majority voting. Finally, we improved the overall efficiency of these algorithms using this ensemble approach. Overview of approach is given below as show in 7.

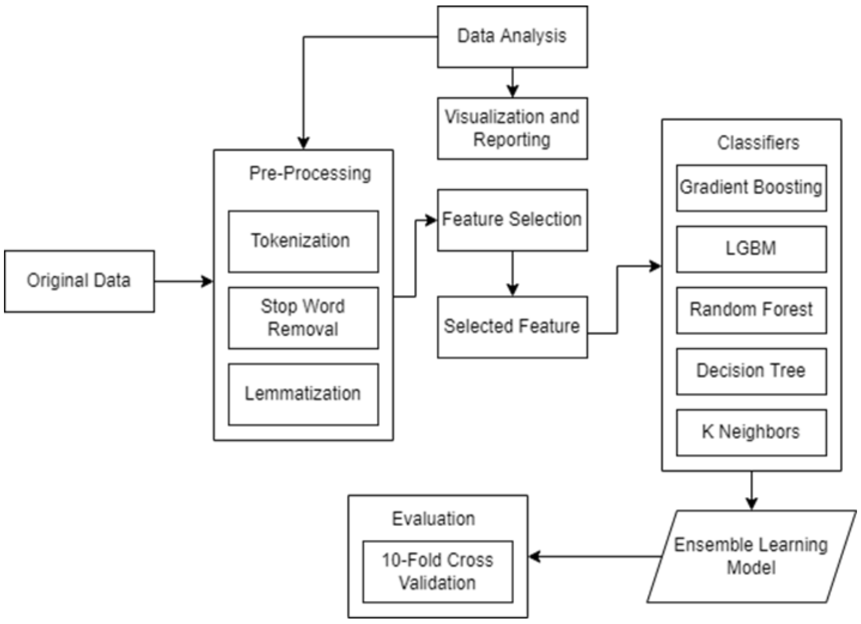


Fig. 7. Proposed Approach

4.1. Ensemble Learning

Ensemble learning classifiers are used in our technique to predict app success. Ensemble learning is a broad meta approach to machine learning that looks for the best prediction per- for Mance by combining approaches to get the maximum accuracy. Because the employment of several machine learning algorithms individually may not yield the best results, the combination of algorithms will combine all of the model's strengths and produce a higher accuracy. In this case, we employ the maximum voting approach, with the outcome being the majority vote of all classifiers. We assessed the performance of several ensemble learning classifiers; the algorithms used are as follows: Decision Tree Classifier (DTC), Random Forest Classifier (RF), K – Nearest Neighbor Classification (KNN), Gradient Boosting Classifier (GBC), and Light Gradient Boost- ing Classifier (LGBC), which are briefly explained below. These algorithms are used to generate models compatible with all of the attributes available in the dataset. Models were implemented using Scikit [35] in Python because ensemble learning approaches are meta-algorithms that intel- grate various machine learning approaches into a single predictive model, reducing variation and bias while improving predictions.

4.1.1. Hard Voting

Ensemble Vote Classifier is a meta-classifier for integrating comparable or conceptually separate machine learning classifiers for classification based on majority or plurality vote. [36]. Ensemble learning is generally employed to enhance a model's efficiency or reduce the chance of selecting a poor model. In machine learning, there are several methods of Ensemble in prac-tice. Here we use a hard voting classifier to fit all the classification models. Then it combines the outcomes of the Decision Tree Classifier, Random Forest C l a s s i f i e r , K - Nearest Neighbor classification Gradient Boosting Classifier and Light Gradient Boosting Classifier. The 8 ills-trades how it all works .The max voting method will enhance the classification precision. Let's take three classifiers as (1), Classifier1 (C1), Classifier2 (C2), and Classifier3 (C3) as an example to show how this approach works.

$$\hat{y} = mode C1(x), C2(x), C3(x), \ldots, Cn(x) \qquad (1)$$

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

$$\hat{y} = mode 1, 0, 1 = 1$$

Class1 is the final decision.

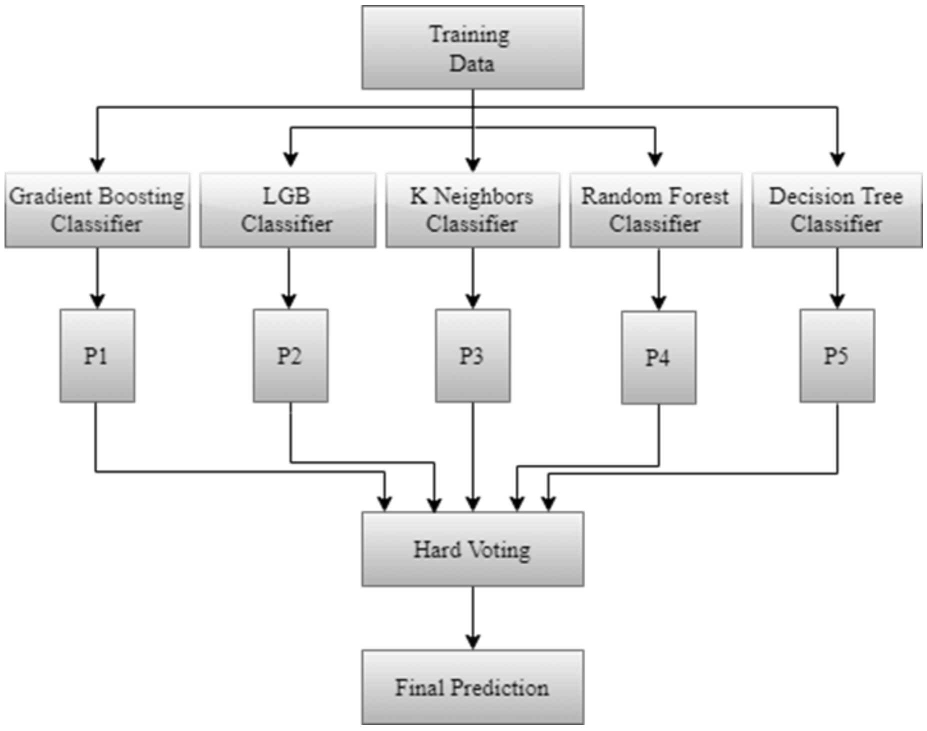


Fig. 8. Ensemble Classifications

5. Results and discussion

5.0.1. Correlation

The heatmap was developed to show the correlations between all of the features with each other features, as shown in 9. As observed from the heatmap, almost all the features are used in-dependently from each other and do not have redundancy. Installs and Reviews have the strongest inverse correlation because more reviews are performed on the most popular apps. Apps with the highest number of installations would generate the highest revenue.

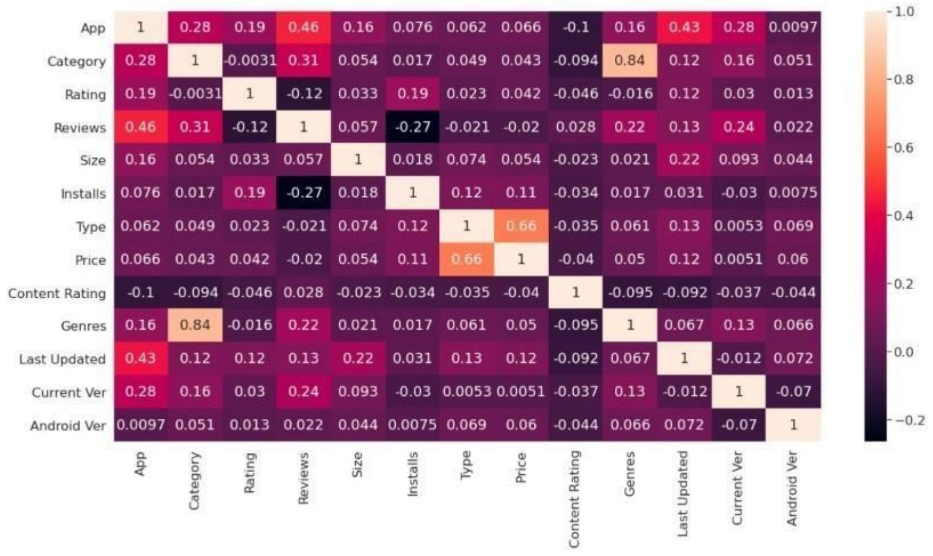


Fig. 9. Heatmap showing relation between features

All the performance measures that are described in section VI are combined in this paper to provide a more comprehensive and accurate picture of each model's performance. 10 depicts the confusion matrix, which demonstrates how each model is evaluated.

All of the different measures used to evaluate a model can be gleaned from the confusion matrix. The Classification report presents a complete list of performance metrics for each target feature category. Similarly, classification reports and confusion matrices are generated for all classifiers. Table I summarizes the findings obtained from the classification reports of all employed models.

When we analyze our results, we see ensemble learning algorithms provide better results than simple machine learning algorithms. The objective Feature in this paper is the number of installs column. The number of categories into which the target Feature is divided significantly im- pacts the models' average accuracy scores. We trained our machine learning algorithms- DT, RF, KNN, GB, and LGB using Google Play apps Dataset obtained from Kaggle. We use a hard voting ensemble-based approach for final prediction. To determine the robustness of each model, we

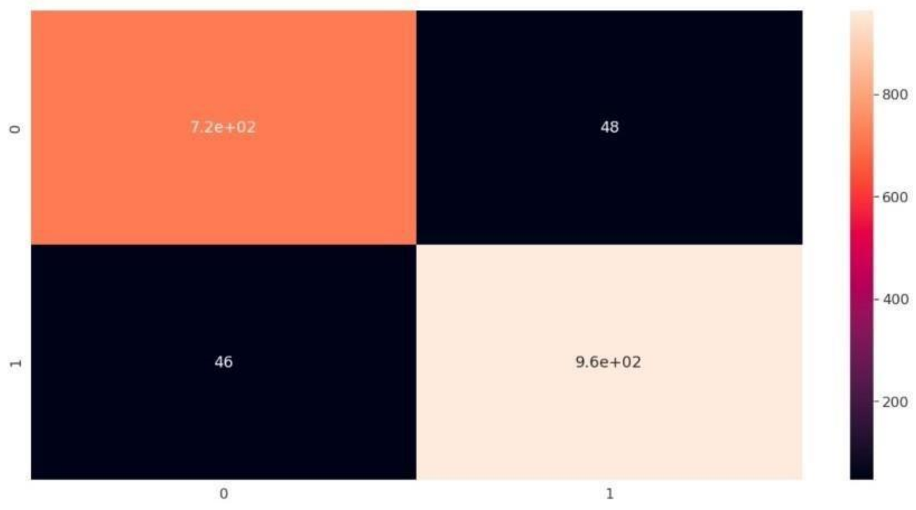


Fig. 10. Confusion Matrix

Table 1. Comparison of Performance Measure for All Models

Classifiers	Accuracy%	Precision	Recall	F1-score
Voting Classifier	96.772239	0.965657	0.946535	0.956000
GB	94.884756	0.963600	0.943564	0.9553477
LGBM	94.730104	0.966734	0.949505	0.958042
Random Forest	94.730005	0.968560	0.945545	0.956914
Decision Tree	94.505187	0.952569	0.954455	0.953511
KNN	83.206968	0.968560	0.956914	0.945545

employed a confusion matrix to determine the accuracy, precision, recall, and f1-score. The av- enrage accuracy of Boating classifier, LGBM RF,DT, and KNN are 94.884756%, 97.772239%, 94.730104%, 94.730005, 94.505187%, and 83.206968% respectively. Table I shows the classic- fiction performance measures of all models. We see that GB and LGB give the outcome more accurately, and KNN gives the least accurate outcome. When the outcomes are combined, hard voting gives an accuracy of 97.772239% (If Including Reviews and Ratings). When not include- Ing the rating and reviews features, the Classifier has around 80% accuracy. If the number of app installs exceeds a specific threshold, the classification models can accurately predict whether an app will be successful or unsuccessful.

6. Evaluation

In this section, we evaluate the proposed approach on the google play store apps collected from Kaggle.

Table 2. Comparison with Previous Approaches

Off-the-Shelf	Algorithms	Dataset	Accuracy
Dehkordi et al.(2020)	Neural Networks	100 successful and 100 unsuccessful Android apps	Without PCA 95.5%, with PCA 99.99%
A. Sing et al. (2020)	Decision Tree Clas- sifier, SVM,Random Forest, XGBoost and KNN	Google Play store	70.49%, 80.34%, 75.59%, 79.99% and 77.26%, respectively
B. T. Magar et al.(2021)	Logistic Regression, Random Forests, K- Nearest Neighbors, Stochastic Gradient Descent, Decision Trees and SVM	Google Play Store apps	0.78476%, 0.71063%, 0.79346%, 0.76386%, 0.77033%, 0.79190% respec- tively
Proposed Model	Ensemble Learning	Google Play store	96.772239%

6.1. Research questions

In the evaluation, the following research questions are being investigated:

- RQ1: Does the proposed ensemble learning approach surpass the existing approaches? ifyes? to what extend?
- RQ2: Does preprocessing step influence the performance of the proposed approach?
- RQ3: Does ensemble learning outperforms other classifiers in predicting app success.

6.1.1. RQ1: Proposed Approach Accuracy

In the first research question (RQ1), the accuracy of the proposed approach is computed. Here we employed an ensemble learning-based approach to predict an app’s success. First, we used training data to train base models— DT, RF, KNN, GB, and LGB. After training, we used test data to evaluate our models’ performance, with each model providing an individual prediction. The predictions of these models are used as additional input for our ensemble learning, which is a composite model trained to provide the final prediction. To make a final prediction, we used a hard voting ensemble-based approach. To determine the robustness of each model, we employed a confusion matrix to determine the accuracy, precision, recall, and f1-score. In the case of App success prediction, the precision should be maximized because which app would be successful must be detected as successful accurately. Evaluation results of — DT, RF, KNN, GB, LGB AND Voting classifier are presented in Table I. The classifiers are listed in the 1st column. Columns 3–5 present the results of each classifier's precision, recall, and f-measure. Each row shows the average performance of a specific classifier. The average precision, re- call, and f-measure of the — LGB, GB, Voting Classifier, RF, DT, and KNN, are (0.966734, 0.949505 and 0.958042) (0.963600%, 0.943564% and 0.9553477%), (0.985657%, 0.966535% and 0.976000%), (0.968560%, 0.945545% and 0.956914%), (0.952569%, 0.954455% and 0.953 511%), and (0.968560%, 0.956914% and 0.945545%,) respectively.

6.1.2. RQ2: Influence of Preprocessing

Apps Data contains worthless and irrelevant information, such as stop words and punctuation (as mentioned in Section 3.2). Providing such data to machine learning algorithms makes the training phase more challenging. It decreases their efficiency and raises the total cost of computing. To answer RQ2, we compare results of proposed approach with and without preprocessing.

Table 3. Influence of Preprocessing

Preprocessing	Accuracy%	Precision	Recall
Disabled			
DT	67.492166	0.693496	0.695382
KNN	58.469450	0.693496	0.943564
Ensemble	95.632687	0.803563	0.82269
Enabled			
DT	94.505187	0.952569	0.954455
KNN	83.206968	0.968560	0.956914
Ensemble	96.772239	0.965657	0.946535
Improvement			
DT	27.013021	0.259073	0.259075
KNN	24.737518	0.338139	0.310033
Ensemble	1.139552	0.162094	0.123845

The findings of the evaluation are presented in Table 3. The input settings for preprocessing are shown in the first column of the table. Columns 2–4 show the accuracy, precision, and recall of performance results. Rows show the proposed approach's performance with different prepare-cussing settings. In the section on improvements, we show how the proposed approach works better with different preprocessing input settings. By enabling preprocessing, the average accu-racy, precision, and recall of the proposed approach are (96.772239%, 0.965657, and 0.946535)., and disabling preprocessing are (95.632687%, 0.803563 and 0.82269). From Table 3, the following are some observations that we make:

• The preprocessing layers used in the proposed approach significantly improve perfor- mance. The evaluation findings indicate that the performance improvement in Accuracy, precision, and recall is 1.139552%, 0.162094, and 0.123845, respectively.

• Disabling preprocessing reduces the precision of the proposed method from 0.965657 to 0.803563. Because using the proposed approach without preprocessing may include un- desired words as features, this may be one of the possible reasons for the decrease in performance. This could impact the proposed approach's efficiency and processing time.

• The above analysis concludes that preprocessing is a crucial step for the proposed approach.

6.1.3. RQ3:

To answer the third research question (RQ3), we compare previous approaches with the proposed approach. Table 2 shows the comparison between the off-the-shelf and proposed approach. When we come to the performances and compare them with the previous approaches that had been done to show the findings and the data collection regarding the use of smartphones in the use of app development and their uses. Lastly, the Proposed Model suggested describing the research prob-lem having the Mobile App Success prediction containing algorithms of Ensemble Learning. This model had the dataset of the Google Play store. And consisting of 97.772239%, 0.985657, 0.966535, 0.976000, Accuracy, Precision, F1-Score, and Recall, respectively. Moreover, this section demonstrates how well our ensemble model performs compared to other studies in this domain. As shown in Table 1, the Android operating system's highest accuracy is 96.772239%, which pertains to the proposed approach. This signifies that our accuracy in the field of Android apps is higher than the accuracy of others on the Android platforms.

6.1.4. Performance Measure

Precision, recall, F1-score, and accuracy are four different measures used to evaluate each model's performance. Although any of these criteria may be insufficient in a defining model when used individually, they provide a comprehensive evaluation of the models when used col-electively.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

Accuracy is the ratio of accurately categorized observations to total observations.

$$Accuracy = \frac{720 + 960}{720 + 960 + 48 + 46} = 0.9470$$

In equation (2), we have TP as True Positives is 720 and TN as True Negatives is 960 with whichwe have FP as False Positive is 48 and FN as False Negative is 46. We got the Accuracy as 0.9470%.

$$Precision = \frac{TP\ TP}{+ FP} \tag{3}$$

The ratio of accurately identified positive observations to total classified positive observations isknown as precision.

$$Precision = \frac{720}{720 + 48} = 0.9375$$

In equation (3), the value of TP is 720 and FP is 48. we got the precision as 0.9375%.

$$Recall = \frac{TP\ TP}{+ FN} \tag{4}$$

The ratio of correctly identified positive observations to all true positive cases is known as Recall.

$$Recall = \frac{720}{720 + 46} = 0.9399$$

In equation (4), the value of TP is 720 and FN is 46. And we get 0.9399% as Recall.

$$F1 = \frac{2 * TP}{2 * TP + FP + FN} \tag{5}$$

F1 score is the average of precision and recall.

$$F1 = \frac{2 * 720}{} = 0.93872 * 720 +$$
$$48 + 46$$

In equation (5), the value of TP is 720 and FP is 48 and FN is 46. And we achieved the F1 score as 0.9387%.
Accuracy is essential when True Positives and True Negatives are crucial, whereas F1-score is used when False Positives and False Negatives are crucial. In this approach, TP represents the number of applications predicted to succeed. TN represents the number of applications that are accurately predicted to fail. FP represents the number of applications that are predicted to suc-ceed incorrectly. FN represents the number of applications that are predicted to fail incorrectly. Where to this approach for successful prediction of apps, TP means the number of apps that are correctly predicted to succeed, whereas TN means the number of apps that are correctly predicted to fail, the number of applications wrongly predicted as success is FP, while the number of apps incorrectly predicted as failure is FN.

7. Conclusion and Future work

The Google Play Store is the world's largest app store. However, it confronts several signif-icant challenges, such as predicting the app's success. Consequently, a large number of Mobile Apps did not receive a satisfactory solution, squandering the developers' time and effort. Sev- erzzal factors influence an app's popularity and success. These factors are diverse and varied, such as app rating, pricing, etc. At this end, we present an ensemble learning-based approach for predicting whether or not a mobile app will be successful. We employ natural language pro- cessing technologies to perform preprocessing. These select features give us optimal results and train an ensemble learning that predicts whether or not a certain app will acquire its solution. As a result of this, developers will be able to save time and effort. Data collected from Kaggle is used to evaluate the proposed approach. The average accuracy, recall, and f-measure of the voting classifier, according to hold-out validation, are up to 0.985657 0.966535 and 0.976000 respectively. The significance of our research is to demonstrate the app requirement description helps in their success prediction. In the future, additional attributes would be helpful to increase classification capabilities. These attributes may include internal app factors and performance. Furthermore, using more advanced algorithms, for example, Deep Neural networks, may lead to better classifications, allowing application developers to improve the success rates of their apps.

References

[1] T. DENG S. KANTHAWALA J. MENG W. PENG A. KONONOVA Q. HAO Q. ZHANG and P. DAVID, *Measuring smartphone usage and task switching with log tracking and self-reports*, Mobile Media Communication, vol. 7, no. 1, pp. 3–23, 2019.

[2] H. HU G. ZHANG X. YANG H. ZHANG L.LEI and P. WANG, *Online gaming addiction and de- pressive symptoms among game players of the glory of the king in china: the mediating role of affect balance and the moderating role of flow experience* International Journal of Mental Health and Ad- diction, pp. 1–14, 2021

[3] S. COMINO F. M.MANENTI and F. MARIUZZO,*Updates management in mobile applications: itunes versus google play*, Journal of Economics Management Strategy,

vol. 28, no. 3, pp. 392–419,2019.

[4] M. R. ISLAM,*Numeric rating of apps on google play store by sentiment analysis on user reviews*, in2014 International Conference on Electrical Engineering and Information Communication Technol- ogy. IEEE, 2014, pp. 1–4.

[5] B. T. MAGAR S. MALI and E. ABDELFATTAH, *App success classification using machine learn- ing models*, in 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC). IEEE, 2021, pp. 0642–0647.

[6] W. LUIZ F. VIEGAS R. ALENCAR F.MOURÃ O T. SALLES D.CARVALHO M. A.GONC¸ ALVES and L. ROCHA,*A feature-oriented sentiment rating for mobile app reviews*, in Proceedings of the 2018 World Wide Web Conference, 2018, pp. 1909–1918.

[7] G. M. M.BASHIR M. S.HOSSEN D. KARMOKER and M. J. KAMAL, *Android apps success pre- diction before uploading on google play store*, in 2019 International Conference on Sustainable Tech- nologies for Industry 4.0 (STI). IEEE, 2019, pp. 1–6.

[8] X. XIA E. SHIHAB Y. KAMEI D.LO and X.WANG,*Predicting crashing releases of mobile applica- tions*, in Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engi- neering and Measurement, 2016, pp. 1–10

[9] G.LEE and T. S.RAGHU, *Determinants of mobile apps' success: Evidence from the app store market*, Journal of Management Information Systems, vol. 31, no. 2, pp. 133–170, 2014.

[10] H. M.GOMES J. P. BARDDAL F.ENEMBRECK and A. BIFET *A survey on ensemble learning for data stream classification*, ACM Computing Surveys (CSUR), vol. 50, no. 2, pp. 1–36, 2017.

[11] G.TU¨ Y S Ü ZO GLU and D.Birant, *Enhanced bagging (ebagging): A novel approach for ensemble learning*, International Arab Journal of Information Technology, vol. 17, no. 4, 2020.

[12] Y. WANG D.WANG N. GENG Y. WANG Y. YIN and Y.JIN,*Stacking- based ensemble learning of de- cision trees for interpretable prostate cancer detection*, Applied Soft Computing, vol. 77, pp. 188–204, 2019.

[13] N. BRUKHIM E. HAZAN and K. SINGH , *A boosting approach to reinforcement learning*, arXiv preprint arXiv:2108.09767, 2021.

[14] A. SING D. TYAGI B. YADAV ET AL.,*Mobile app success prediction*, International Journal for Research in Applied Science Engineering Technology (IJRASET), vol. 8, no. 6, pp. 1674–1679, 2020.

[15] V.DIBIA and C. WAGNER, *Success within app distribution platforms: The contribution of app di- versity and app cohesivity*, in 2015 48th Hawaii International Conference on System Sciences. IEEE,2015, pp. 4304–4313.

[16] G. C.-C. SHEN, *Users' adoption of mobile applications: Product type and message framing's mod- erating effect*, Journal of Business Research, vol. 68, no. 11, pp. 2317–2321, 2015.

[17] W. N. PICOTO R.DUARTE and I.PINTO,*Uncovering top-ranking factors for mobile apps through amultimethod approach*, Journal of Business Research, vol. 101, pp. 668–674, 2019.

[18] C. CHATFIELD and A. J. COLLINS, *Introduction to multivariate analysis*, Routledge, 2018.

[19] A. SING, D. TYAGI, and B. YADAV, *Mobile App Success Prediction* International Journal for Re- search in Applied Science Engineering Technology (IJRASET), vol. 8, no. 6, pp. 1674-1679, 2020.

[20] A. MUEEZ K. AHMED T. ISLAM and W. IQBAL,*Exploratory data analysis and success prediction of google play store apps*, Ph.D. dissertation, BRAC University, 2018.

[21] M.-Y. DAY and Y.-D. LIN, *Deep learning for sentiment analysis on google play consumer review* , in 2017 IEEE international conference on information reuse and integration (IRI). IEEE, 2017, pp. 382–388.

[22] M. R.DEHKORDI H. SEIFZADEH G. BEYDOUN and M. H. NADIMI-SHAHRAKI, *Success pre- diction of android applications in a novel repository using*

neural networks, Complex Intelligent Systems, vol. 6, no. 3, pp. 573– 590, 2020.

[23] M. SULEMAN A. MALIK and S. S. HUSSAIN,*Google play store app ranking prediction using machine learning algorithm*, Urdu News Headline, Text Classification by Using Different Machine Learning Algorithms, vol. 57, 2019.

[24] I. PORTUGAL P. ALENCAR and D. COWAN, *The use of machine learning algorithms in recom- mender systems: A systematic review*, Expert Systems with Applications, vol. 97, pp. 205–227, 2018.

[25] M. CRISTOFARO, *E-business evolution: an analysis of mobile applications' business models*, Tech- nology Analysis Strategic Management, vol. 32, no. 1, pp. 88–103, 2020.

[26] R. ARALIKATTE G. SRIDHARA N. GANTAYAT and S. MANI, *Fault in your stars: an analysis of android app reviews*, in Proceedings of the acm india joint international conference on data science and management of data, 2018, pp. 57–66.

[27] S. SADIQ M. UMER S. ULLAH S. MIRJALILI V. RUPAPARA and M. NAPPI, *Discrepancy detec- tion between actual user reviews and numeric ratings of google app store using deep learning*, ExpertSystems with Applications, vol. 181, p. 115111, 2021.

[28] Y. YAO W. X. ZHAO Y. WANG H. TONG F. XU and J. LU, *Version-aware rating prediction for mobile app recommendation*, ACM Transactions on Information Systems (TOIS), vol. 35, no. 4, pp. 1–33, 2017.

[29] M. HARMAN Y. JIA and Y. ZHANG,*App store mining and analysis: Msr for app stores*, in 2012 9th IEEE working conference on mining software repositories (MSR). IEEE, 2012, pp. 108–111.

[30] B.FU J. LIN L. LI C. FALOUTSOS J. HONG and N. SADEH, *Why people hate your app: Making sense of user feedback in a mobile app store*, in Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, 2013, pp. 1276–1284.

[31] R. MIHALCEA and P. TARAU, *Textrank: Bringing order into text*, in Proceedings of the 2004 con- ference on empirical methods in natural language processing, 2004, pp. 404–411.

[32] B. LIU, *Sentiment analysis and opinion mining*, Synthesis lectures on human language technologies,vol. 5, no. 1, pp. 1–167, 2012.

[33] H. ZHU C. LIU Y. GE H. XIONG and E. CHEN,*Popularity modeling for mobile apps: A sequential approach*, IEEE transactions on cybernetics, vol. 45, no. 7, pp. 1303–1314, 2014.

[34] Y.WANG N. J. YUAN Y. SUN C.QIN and X.XIE, *App download forecasting: An evolutionary hier- archical competition approach.*, in IJCAI, 2017, pp. 2978–2984.

[35] G. HACKELING, *Mastering Machine Learning with scikit-learn*. Packt Publishing Ltd, 2017.

[36] S. RASCHKA, *Ensemblevoteclassifier,* rasbt. github. io/mlxtend/user/classifier/EnsembleVoteClassifir,2014.