Received: 26 September 2019 | Revised: 24 January 2020 | Accepted: 2 March 2020

DOI: 10.4218/etrij.2019-0443

ORIGINAL ARTICLE

ETRI Journal WILEY

Predicting numeric ratings for Google apps using text features and ensemble learning

Muhammad Umer[1] | **Imran Ashraf**[2] | **Arif Mehmood**[3] | **Saleem Ullah**[1] | **Gyu Sang Choi**[2]

[1]Department of Computer Science, Khawaja Freed University, Punjab, Pakistan

[2]Department of Information and Communication Engineering, Yeungnam Univeristy, Gyeongsan, Rep. of Korea

[3]Department of Computer Science and Information Technology, The Islamia University of Bahawalpur, Punjab, Pakistan

Correspondence
Gyu Sang Choi, Department of Information and Communication Engineering, Yeungnam University, Gyeongsan, Korea.
Email: castchoi@ynu.ac.kr

Saleem Ullah, Department of Computer Science, Khawaja Freed University, Punjab, Paksitan.
Email: saleem.ullah@kfueit.edu.pk

Funding information
This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2019R1A2C1006159), and MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2019-2016-0-00313) supervised by the IITP (Institute for Information & Communications Technology Promotion).

Application (app) ratings are feedback provided voluntarily by users and serve as important evaluation criteria for apps. However, these ratings can often be biased owing to insufficient or missing votes. Additionally, significant differences have been observed between numeric ratings and user reviews. This study aims to predict the numeric ratings of Google apps using machine learning classifiers. It exploits numeric app ratings provided by users as training data and returns authentic mobile app ratings by analyzing user reviews. An ensemble learning model is proposed for this purpose that considers term frequency/inverse document frequency (TF/IDF) features. Three TF/IDF features, including unigrams, bigrams, and trigrams, were used. The dataset was scraped from the Google Play store, extracting data from 14 different app categories. Biased and unbiased user ratings were discriminated using TextBlob analysis to formulate the ground truth, from which the classifier prediction accuracy was then evaluated. The results demonstrate the high potential for machine learning-based classifiers to predict authentic numeric ratings based on actual user reviews.

KEYWORDS
data mining, ensemble learning, Google app rating, opinion mining, text features, text mining

1 | INTRODUCTION

Opinion mining is the process of identifying and detecting subjective information using natural language processing, text analysis, and computational linguistics. It is a subfield of text mining, which aims at building systems that identify and extract opinions within a text. Opinions play a significant role in decision-making in daily life. The ubiquity of

Muhammad Umer and Imran Ashraf contributed equally.

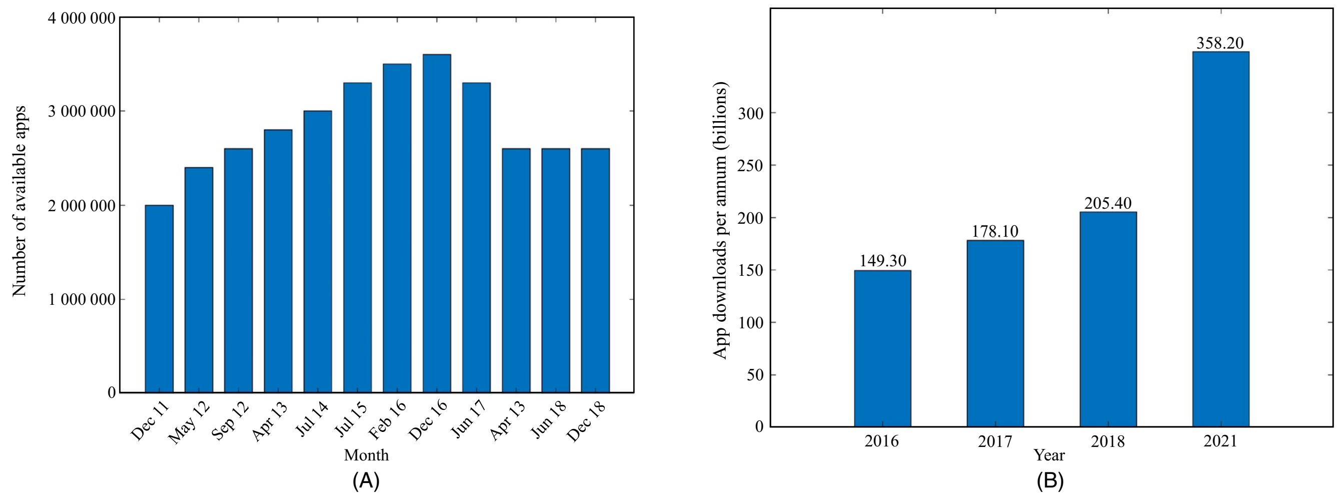
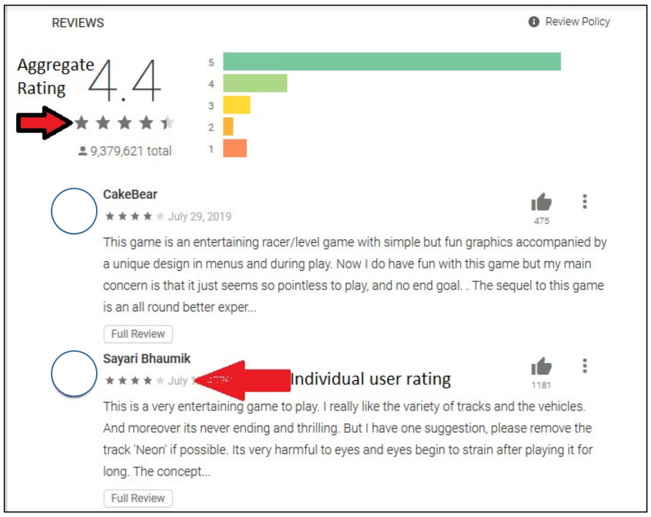


FIGURE 1 Evolution of apps on the Google Play store [1,2]: (A) Total apps on the Google Play Store, (B) Total apps downloaded from the Google Play Store

smartphones increasingly impacts our work- and lifestyles. Hundreds of thousands of applications ("apps") provide a multitude of everyday services encompassing medical care, fitness, beauty, surveillance, sports, etc. As of March 2019, a total of 2.6 million apps are available on the Google Play store [1], as shown in Figure 1A. Millions of users can download, use, and evaluate these apps and provide feedback in the form of reviews and numeric ratings. Currently, 205.4 billion apps are downloaded annually and this number is expected to reach 352.9 billion by 2021 [2], as shown in Figure 1B. User reviews and numeric ratings are the leading guiding factors for other users considering whether to adopt particular apps. Research [3] has demonstrated the strong influence of user reviews and numeric ratings in spreading the use of mobile apps generally. It was shown that consumers can prefer to buy a 5-star- rather than a 4-star-rated product, even at an additional cost of 20%–99%. The ability to share ratings, bug reports, and reviews increases the scope of engagement of users and app developers alike [4].

User feedback is provided in two forms, text review and numeric ratings, as illustrated in Figure 2. A text review, on the one hand, may contain positive or negative comments submitted by a user about a specific app or policy. These data can be highly useful for performing marketing analysis, managing public relations, conducting product reviews, net promoter scoring, giving product feedback, delivering customer service, and so on [5].

A numeric rating, on the other hand, is a quantitative value given by a user, ranging from 1 to 5. It is an aggregated rating, typically defined by an app user selecting one or more star icons. A high numeric rating motivates more potential users to download the app. Given the importance of reviews and app ratings, the possibility of biased and fake reviews is a substantial concern. Currently, there is no standard mechanism for validating the authenticity of numeric


REVIEWS

Review Policy

Aggregate Rating 4.4

★★★★☆

9,379,621 total

5	
4	
3	
2	
1	

CakeBear
★★★★☆ July 29, 2019
475
This game is an entertaining racer/level game with simple but fun graphics accompanied by a unique design in menus and during play. Now I do have fun with this game but my main concern is that it just seems so pointless to play, and no end goal. . The sequel to this game is an all round better exper...
Full Review

Sayari Bhaumik
★★★★☆ July
Individual user rating
1181
This is a very entertaining game to play. I really like the variety of tracks and the vehicles. And moreover its never ending and thrilling. But I have one suggestion, please remove the track 'Neon' if possible. Its very harmful to eyes and eyes begin to strain after playing it for long. The concept...
Full Review

FIGURE 2 The Google Play store apps review

user ratings. This causes inconvenience and uncertainty for end users wishing to select the most suitable mobile apps. Predominantly, users tend to select an app based only on the first comments and reviews they read. Typically, time-pressured users feel they do not have the leisure to read entire reviews, which results in their selecting the wrong app.

Although several approaches to address these problems have been proposed [6,7], they focus essentially on identifying only a polarity inherent in user reviews. Polarity and subjectivity are defined in terms of three descriptors: *positive*, *negative*, and *neutral*. These approaches focus on reviews exclusively without considering actual app ratings, thereby ignoring the problem of mismatched numeric ratings. Our experiments show that displayed numeric ratings and user reviews are often mismatched, resulting in contradictory evaluations.

This proposes an approach designed to overcome this shortcoming by predicting the numeric rating for a mobile app based on user reviews. A machine learning-based ensemble classifier is used for this purpose. The dataset is first preprocessed by applying a *case transformation*, *removing numbers*, *removing stop words*, *tokenization*, and *stemming*. To balance the dataset and avoid training classifiers disproportionately on 5-star-rated instances, we extracted 25 000 records associated with each target rating to train our ensemble learning classifiers on all the text features amenable to rating. Hence, the new dataset contained 125 000 instances. We then applied the vector-space modeling (VSM) techniques of term frequency/inverse document frequency (TF/IDF), considering unigrams, bigrams, and trigrams and TF, to the preprocessed data. Features generated by these techniques were then passed to ensemble learning models. The key contributions of this research can be summarized as follows:

- Machine learning algorithms, including the random forest (RF), the gradient boosting classifier (GBM), the extreme gradient boosting classifier (XGB), the AdaBoost classifier (AB), and the extra tree classifier (ET) were applied to predict numeric ratings.
- The classifier accuracy was analyzed from the following two perspectives: (a) using the textual features derived from the reviews alone, and (b) using both the textual features and the emoticons available in the reviews.
- For validation, famous apps belonging to each category were used to compare the numeric ratings predicted by ensemble learning with user's actual ratings.

The rest of the paper is organized as follows. Section 2 reviews prior-related research. Section 3 introduces the background information on the ensemble learning classifiers used in this study. Section 4 presents our proposed approach and the experimental dataset. The results are discussed in Section 5 and concluding remarks are given in Section 6.

2 | RELATED WORK

There has been a constant growth in the public and private information stored within the internet. This includes textual data expressing people's opinions on review sites, forums, blogs, and other social media platforms. Review-based prediction systems allow this unstructured information to be automatically transformed into structured data reflecting public opinion. These structured data can be used subsequently as a measure of users' sentiments about specific applications, products, services, and brands. They can hence provide important information for product and services refinement. This kind of sentiment analysis was conducted in the following studies.

Kumari and others [8] and other researchers [9,10,5] used the naïve Bayes (NB) classifier to classify opinions as positive, negative, or neutral. Wang and others [11] argued that a rating is not entirely determined by a review content. For example, a user may well intend to give a positive review by employing positive words, and yet issue a comparatively lower rating.

Dave and others [12] proposed a method for extracting the polarity in user reviews of products, expressed as poor, mixed, or good. The classifier used was NB. According to Pang et al., although machine learning approaches perform much better for traditional topic-based categorization, they are less successful for sentiment analysis [13].

Information-extraction technologies have also been explored to identify and organize opinions contained in text. For example, some authors [14] proposed a scheme for annotating a low-level representation of opinions within a text. Additionally, they described an opinion-oriented "scenario template" that summarizes the opinions expressed in a document. This approach is helpful for tasks that involve posing question from multiple perspectives. Other authors [15] suggested adopting a statistical analysis based on a spin model, to extract the semantic orientations of words. Mean-field approximations were used to compute the approximate probability in the spin model. Semantic orientations are then evaluated as desirable or undesirable. A smaller number of seed words for the proposed model produce highly accurate semantic orientations based on the English lexicon.

Various sentiment analysis methods have been performed to summarize the ensembles of comments and reviews [16]. These methods use mathematical and statistical methods (especially involving Gaussian distributions) to overcome the problems encountered in sentiment analysis. Although these authors proposed a model, it was not implemented. A recent study [17] investigated the application of a machine learning algorithm to a dataset covering, for example, the app category, the numbers of reviews and downloads, the size, type, and Android version of an app, and the content rating, to predict a Google app ranking. Decision trees, linear regression, logistic regression, support-vector machine, NB classifiers, *k*-means clustering, *k*-nearest neighbors, and artificial neural networks were studied for that purpose.

App ratings have been predicted based on the features provided for app [18,19]. Experiments were performed on the BlackBerry World and Samsung Android stores to collect the raw features provided for the apps, including their price, rank of downloads, ratings, and textual descriptions. The features were then encoded into a numerical vector to be used in case-based reasoning and to predict the app rating. In contrast to the above-cited studies, other authors [20] investigated the nature of sentiments expressed in Google app reviews. Their study measured opinions and sentiments represented in user reviews through a variety of emojis expressing, for example, negativity, positivity, anger, or excitement. It evaluated whether those sentiments are informative for the purpose of app development and refinement.

However, the above studies are unsatisfactory in various respects and are unsuitable for predicting numeric ratings of Google apps. First, text-mining techniques are ineffective when applied to app reviews, as it has Unicode supported language with a limited number of words. Second, those studies are based either on rating predictions made using inherent app features or on external features (eg, price, bug report, etc.). None of those studies investigated the possible discrepancies between users' numeric ratings and reviews. To our knowledge, this study is the first to investigate such discrepancies and to base numeric-rating predictions for Google apps on users' reviews.

3 | ENSEMBLE LEARNING METHODS USED FOR PREDICTION

Our approach exploits ensemble learning classifiers to predict numeric ratings based on users' reviews of mobile apps. We evaluated the performance of various ensemble learning classifiers, discussed briefly below. Models were implemented using *Scikit learn* [21,22] in Python. Since ensemble learning methods are meta-algorithms that combine several machine learning techniques into one predictive model, they help to decrease the variance (bagging) and bias (boosting) and improve predictions (stacking) [23].

There is a copious literature on the lexicon- and machine learning-based classification of unstructured and structured data. However, machine learning-based classifiers are preferred over other methods on account of their superior performance. For example, some authors [24] compared the performance of 10 models (five each from lexicon- and machine learning-based classifiers) in terms of their classification efficacy. They concluded that RF and NB classifiers are substantially better at correctly uncovering human intuition. In contrast, lexicon-based approaches perform worse than machine learning. These findings were corroborated by other authors who concluded that machine learning-based classifiers outperform lexicon-based approaches [25,26]. In light of other studies [23,25,26], the present study uses machine learning-based classifiers to predict Google app ratings.

RF provides an alternative to decision trees. It reduces the variance by generating several trees; however, the results are not easily interpretable. It is used for classification and regression. It involves an ensemble of decision trees in which results are aggregated into one final result. RF reduces the variance in two ways: first by training on different data samples and second by using a random subset of features [27].

AdaBoost (AB), a short form for adaptive boosting, is the first boosting algorithm to outperform other boosting algorithms in classification tasks in terms of accuracy. AdaBoost is adaptive in the sense that weak learners are adapted by considering earlier misclassifications by previous classifiers [28]. Any individual learner can be weak. However, when considering a complete learned model, AdaBoost achieves a high accuracy. It is also used for finding important features by calculating a feature importance score.

Gradient Boosting (GB) is an ensemble learning technique used for regression and classification problems. It produces a prediction model consisting of an ensemble of weak prediction models, typically in the form of decision trees [29]. Boosting is a method for converting weak learners into strong learners. In boosting, each new tree is fitted to a modified version of the original dataset. The good performance of GB stems from its use of gradients in the loss function, which measures how efficiently the model coefficients fit the underlying data. The precise definition of the loss function depends on which quantity, feature, or property is to be optimized.

Extreme gradient boosting (XGBoost) is a boosting algorithm with much appeal among data scientists for its execution speed and performance. Its good performance stems from its implementation, which is composed of three components, namely gradient boosting, stochastic gradient boosting, and regularized gradient boosting. It is used for both classification and regression problems. XGBoost attracted attention when it earned success to several Kaggle competition winners [30]. Feature importance scores are also calculated, as with other boosting and bagging classifiers.

The extra-trees (ET) classifier implements a meta-estimator that fits a number of randomized decision trees (ET) on various subsamples of a dataset, performs averaging to improve the predictive accuracy, and controls overfitting [31]. Unlike RF, the entire sample is used by the ET classifier at each step and decision boundaries are picked randomly, rather than by selecting the best one. It is also known as "extremely randomized tree."

4 | PROPOSED APPROACH FOR EVALUATING APPLICATION RATINGS

This section describes the proposed approach, its modules, and the dataset used in the experiment.

The architecture of the proposed approach for predicting numeric ratings is outlined in Figure 3. It involves several subtasks, described separately below.

4.1 | Dataset

The Google apps dataset was scraped from the Google Play store using the *BeautifulSoup* web scraper. The data were scraped for applications released no later than 2014 to ensure a minimum time span of 5 years. The following criteria were applied:

- The app must be 5 years old at least.
- The app must have at least 4000 posted reviews.

The dataset contains a total of 502 658 records and includes the following attributes, described in Table 1: App_ category, App_ name, App_ id, App_ review, and App_ rating. Although other features and related meta-data can be scraped from the Google Play store in principle, this study focused on analyzing the discrepancy between user reviews and app-rating predictions based on the features extracted from the reviews. We therefore neglected other meta-data that were irrelevant.

The scraped data consist of 14 different categories of mobile apps. The primary motivation for sampling over multiple categories was to cover the broad range of reviews available. Each category (eg, sports, news, entertainment, health, and business) contains different types of textual reviews displaying a variety of expressions and words. Our purpose was therefore to evaluate the performance of the classifiers by applying them to diverse reviews, rather than considering only a few categories, to properly assess the classification accuracy. Table 2 shows a sample of reviews obtained from the photography, health and fitness, finance, weather, and medical categories.

Figure 4 shows the names and relative size of all the categories in the dataset. Each app in the scraped database has at least 4000 reviews. The App_ rating and App_ review data from individual users were analyzed. The data were scraped using the *BeautifulSoup* (BS) web scraper, as depicted in the flow chart in Figure 5.

Data visualization is essential for understanding the dataset by identifying patterns, trends, and correlations that might go undetected in text-based data. In several datasets, such connections can be recognized conveniently through data visualization. Therefore, we visualized the dataset to display the numeric ratings given to mobile apps.

Figure 6 identifies 5 as the most frequent app user rating. However, the possibility of biased or even fake ratings given by anonymous users is a major concern. We therefore considered the frequency of numeric ratings in each category. Figure 7 shows that mobile apps in the *casual* category always score higher ratings and are most likely to display a 5-star rating from app users. In contrast, developers of *Video Players & Editors* tend to receive the lowest numeric ratings.

4.2 | Date preprocessing

The dataset collected from the Google Play store is semi-structured or unstructured and contains significant superfluous data (defined as not contributing significantly to the prediction process). Since large datasets require longer training times, and because "stop words" reduce the prediction accuracy, text preprocessing is therefore required to overcome this limitation. Preprocessing involves various tasks including stemming, lowercase conversion, punctuation, and excluding terms which do not carry higher

TABLE 1 Description of attributes in the used dataset

Attribute	Description
App_ category	App category in the Google Play store
App_ name	Actual app name in the Google Play store
App_ id	Unique app_ id assigned in the Google Play store for identification
App_ review	Review given by each individual user for a specific app
App_ rating	Rating given for the app by an individual user.

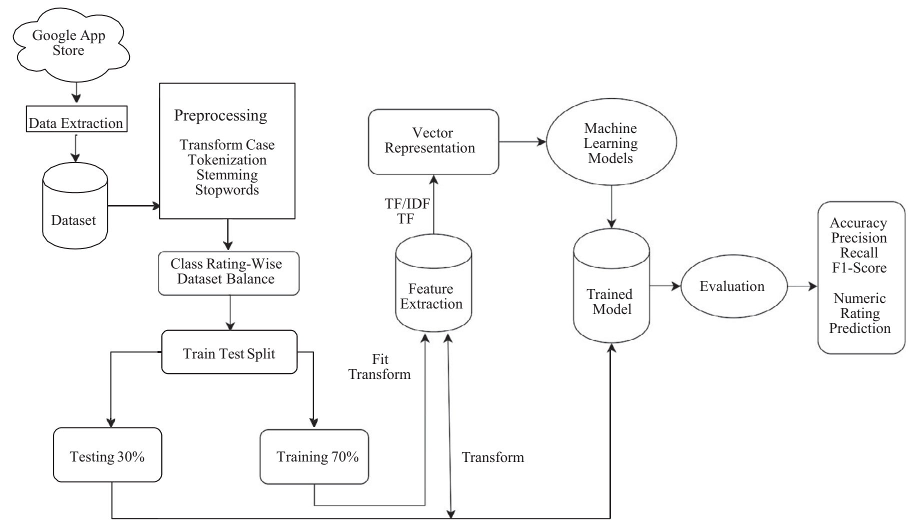


FIGURE 3 Architecture of the proposed approach

Category	Reviews
Photography	'I TOOK SOME PICTURES AND IT HAS A GOOD SNAP I HAVEN'T TRIED THE VIDEO RECORDER YET BUT SURE IT'S GOOD AS THE CAMERA KEEP ON BANGING'
Health & Fitness	'I loved this App,Really All the Exercises are Effective and Awesome to do…Also The Different features like Time, Day Tracker Helps a lot… I just have one query… Will These Exercises affect my height ? I am worried about my Height…I want it to increase.'
Finance	'ATM withdrawal feature via QR codes is broken on Google Pixel due to the app using a wrong aspect ratio with the camera – a square QR code is captured as 2:1, making it not recognizable. Please fix, if you could. Thanks.'
Weather	'Awesome app. Lots of information. Thunderstorms, live radar, hurricane Tracking. 5 day forecast. Everything you need to know about the weather.'
Medical	'I am a Physiotherapist and I need to make my clients understand about the musculoskeletal system and the part that is involved and I always wished if such things (apps) will ever be made. Thank to the creator of this app.'

TABLE 2 Actual reviews from various app categories

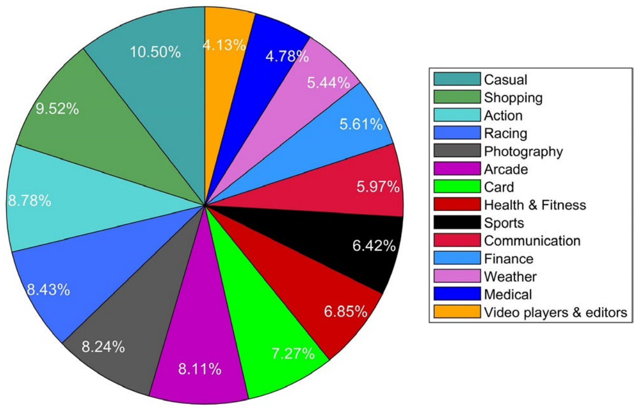


FIGURE 4 Count of each category in the dataset

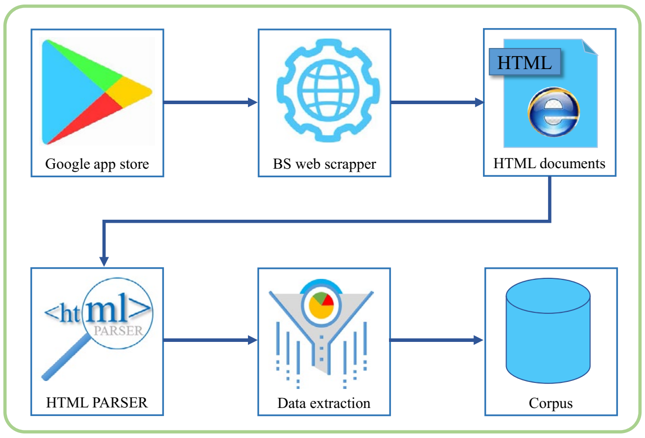


FIGURE 5 Flow diagram for data scraping

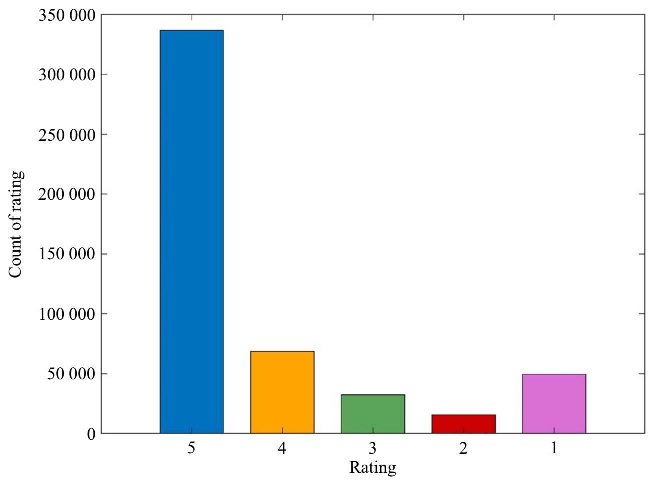


FIGURE 6 Frequency of user ratings

weight in context to the text. Research has shown that text preprocessing plays a significant role in improving the prediction accuracy [32].

The data must therefore be normalized before starting the training process using the proposed approach. To better understand text preprocessing, consider the example of a review for the mobile app *Seven minute workout challenge*. A series of preprocessing steps and their output are shown in Table 3. After the completion of preprocessing, the ensemble learning classifiers can be applied to the processed dataset.

4.3 | Feature selection

Training a supervised machine learning algorithm requires textual documents to be represented in vectorial form. For this purpose, textual data must be converted into numbers

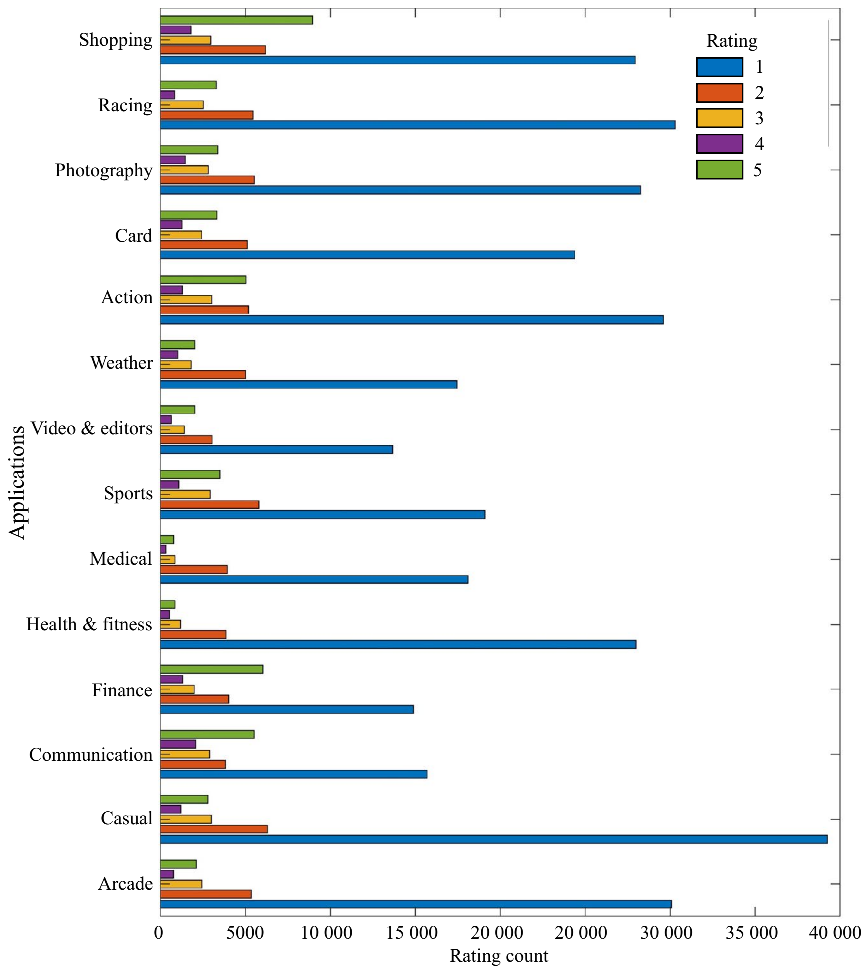


FIGURE 7 User ratings by category

without losing information. This transformation can be achieved using several techniques, for example, Bag-Of-Words (BOW), which assigns a number to each word. However, character-length limitations in reviews can reduced the efficiency of BOW. Insufficient word occurrences can also limit the accuracy of BOW-based approaches [33]. We therefore exploit the term frequency (TF) to perform the transformation [34]. This involves counting the words in a document to produce a matrix displaying the total number of occurrences of each word in the document. As an example of TF, consider a sample review for the app *My Talking Tom*. The original review is:

It is great, my kids love to play it. Thank you for having this game on game store.

After preprocessing, the review becomes *great kid love play. thank game game store*. The matrix obtained by applying TF is shown in Table 4.

We also used TF/IDF (for unigrams, bigrams, and trigrams) to perform feature selection in our proposed approach [35]. TF/IDF assigns a lesser weighting to highly common words (occurring in almost all the documents), but gives a stronger weighting to the words that appear more specifically in a subset of documents. Commonly occurring words are thus assigned a lower weight, while some rare words occurring more specifically in a particular document are considered more important. The outcome of TF/IDF on the above example is shown in Table 5.

4.4 | Accuracy measures

We used different parameters for performance evaluation. Four basic notions are important for understanding the different accuracy parameters of a classifier [36,37].

True Positives (TP): positive predictions about a class which are correctly predicted by the classifier.

True Negatives (TN): negative predictions about a class which are correctly labeled by the classifier.

False Positives (FP): negative instances of a class which are incorrectly labeled as positive by the classifier.

False Negatives (FN): positive instances of a class which are incorrectly labeled as negative by the classifier.

Accuracy is an evaluation parameter widely used in classification models. In terms of TP, TN, FP, and FN, defined above, it is calculated as follows:

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \times 100. \tag{1}$$

Preprocessing steps	Output
Actual text	'\ u003c/\ u003e' I honestly preferred the older version, but the new one isn't to bad. The demonstrator keeps glitching out, and the next exercise keeps popping up while I'm doing a different one.'
Transform to lower case	'\ u003c/\ u003e' i honestly preferred the older version, but the new one isnt to bad. the demonstrator keeps glitching out, and the next exercise keeps popping up while im doing a different one.'
Remov HTML tags	i honestly preferred the older version, but the new one isnt to bad. the demonstrator keeps glitching out, and the next exercise keeps popping up while im doing a different one.'
Remove stop words & tokenization	['honestly', 'preferred', 'older', 'version', ',', 'new', 'one', 'isnt', 'bad', '.', 'demonstrator', 'keeps', 'glitching', ',', 'next', 'exercise', 'keeps', 'popping', 'im', 'different', 'one', '.']
Stemming	'honestly prefer older version, new one isn't bad. demonstrate keep glitch, next exercise keep pop im differ one.'

TABLE 3 Attributes of the used dataset

TABLE 4 Features obtained by applying TF

Game	Great	Kid	love	Play	Store	Thank
2	1	1	1	1	1	1

TABLE 5 Features obtained by applying TF/IDF

Game	Great	Kid	love	Play	Store	Thank
0.63	0.31	0.31	0.31	0.31	0.31	0.31

Precision, in contrast, specifies the percentage of all instances of a class that are correctly labeled as positive, that is,

$$\text{precision} = \frac{\text{TP}}{\text{TP}+\text{FP}}. \tag{2}$$

Recall is often considered as a measure of completeness and represents the proportion of true positive instances of a class that are labeled as such, that is,

$$\text{recall} = \frac{\text{TP}}{\text{TP}+\text{FN}}. \tag{3}$$

The F score, which takes values between 0 and 1, considers both the precision and the recall. It represents the average effect of the precision and recall, calculated as

$$F_1 = 2.\frac{\text{precision.recall}}{\text{precision} + \text{recall}}. \tag{4}$$

TABLE 6 Categories and apps selected for the experiment

App category	Selected app	Total reviews
Sports	Billiards City	32 280
Communication	UC-Browser	30 000
Action	Gun Shot	44 141
Arcade	Tempe Run 2	40 751
Video players & editors	MX Player	20 781
Weather	Weather & Clock widget	27 324
Card	Teen Pati Gold	36 520
Photography	B612	41 440
Shopping	Flipkart	47 840
Health & fitness	Seven	34 415
Finance	PhonePe	28 220
Casual	Candy Crush Saga	52 560
Medical	Pharmapedia Pakistan	24 002
Racing	Beach Buggy Racing	42 384

5 | EXPERIMENT AND RESULTS

This section details the experiment conducted herein, and discusses the results. To predict authentic numeric rating using ensemble learning models, 14 app categories, each containing 12 apps, were selected, giving a dataset with reviews from 168 apps in total. The app categories were selected based on their popularity, as represented by their corresponding rating on the Google Play store. In contrast, one mobile app with a higher number of reviews and with the highest collective rating was selected from each category. Table 6 shows the

TABLE 7 Examples of biased app ratings

User review	User app rating
'It's little waste of time, good just time pass, many ads are coming and irritated me'	4
'Hate the bimbo ad. I don't care for ads with women shaking their a$$'	5
'Fun game. Too bad there is a Kyrsten Sinema prostitution add after every level. Uninstalling.'	5
'It just takes a long time to load'	4
'Please fix the tool long starting of the game plsss'	4

chosen categories and the apps selected in each category for the experiment. It also specifies the total number of reviews obtained for the selected apps.

5.1 | Methodology adopted to evaluate the prediction performance

Numeric scores given by users in the Google App store may be biased or even exaggerated, simply because high ratings attract more new users. The examples shown in Table 7 suggest that there is a genuine discrepancy between reviews and ratings. However, this hypothesis should be assessed systematically.

This study devises an algorithm that utilizes TextBlob to determine the discrepancy between a review posted by a user and the app rating. The algorithm flowchart is shown in Figure 8. The algorithm was implemented in Python, so that the sentiment property of TextBlob returns a named tuple of the form Sentiment (polarity, subjectivity) [38]. The polarity score is a float between −1.0 and 1.0, while the subjectivity is a float between 0.0 and 1.0, where 0.0 represents a strong objectivity and 1.0 a strong subjectivity. For example:

from textblob import TextBlob

t = TextBlob ("He is a cheater. i hate him ")

t.sentiment shows

Sentiment (polarity = −0.8, subjectivity = 0.9)

In our experiment, we calculated the polarity of every review in the dataset and matched it against its corresponding rating to detect possible bias. If the polarity of the review is less than −0.5 and the rating associated with that review is greater than 3, the user rating is considered to be biased. Otherwise, it is considered to be unbiased.

The algorithm outcomes are shown in Table 8. This study considers the discrepancies between 3-, 4-, and 5-star ratings and their corresponding reviews, in the light of the importance of high rating for guiding new users. The analysis shows that 124 238 rating counts were biased out of a total of 502 658 user ratings, which reveals that 24.7162% of the total ratings for selected app categories are biased and the ground truth for unbiased ratings is 75.2838%.

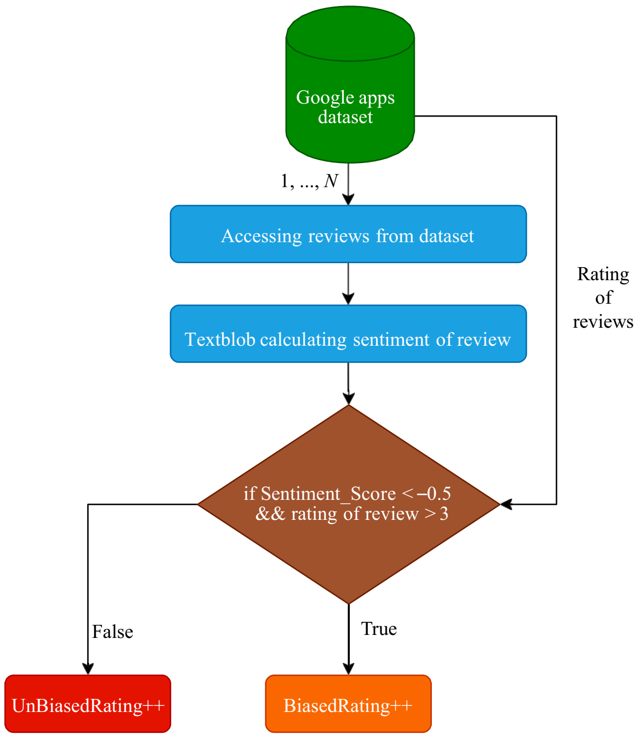

Google apps dataset
1, ..., N
Accessing reviews from dataset
Textblob calculating sentiment of review
Rating of reviews
if Sentiment_Score < −0.5 && rating of review > 3
False
True
UnBiasedRating++
BiasedRating++

FIGURE 8 Flowchart of the method for detecting biased ratings

5.2 | Numeric-rating prediction with user reviews

User reviews were analyzed to predict numeric ratings based on different ensemble learning models and for subsequent comparison against the aggregate rating for the apps considered. An aggregate rating is the original rating given to an app on the Google Play store. The purpose of this comparison is to detect discrepancies between user reviews and ratings. Table 9 shows the numeric ratings predicted by the machine learning classifiers selected for this experiment.

The highest predicted ratings are written in bold and underlined. As shown in Table 9, for 11 out of the 14 categories, ET predicts the highest score. The ratings predicted by ET

TABLE 8 Biased ratings calculated using TextBlob

App rating	Rating count	Biased count	Ground truth
1	49 608	N/A	49 608
2	15 605	N/A	15 605
3	32 238	14 725	17 513
4	68 419	20 479	47 940
5	336 781	89 034	247 747

TABLE 9 Numeric-rating prediction using ensemble classifiers

App Name	App reviews	XGB	RF	GBM	AB	ET	Aggregate rating
Billiards City	4480	4.00	4.21	4.42	3.76	**4.46**	4.47
UC-Browser	3000	3.01	3.27	3.24	2.95	**3.58**	4.20
Gun Shot	3000	3.30	3.43	3.47	3.31	**3.64**	3.69
Temple Run 2	3000	3.85	3.93	3.99	3.74	**4.10**	4.45
MX Player	3000	2.83	2.98	3.06	2.95	**3.29**	3.92
Weather & Clolck Widget	4480	2.74	2.90	**2.95**	2.68	2.93	4.33
Teen Pati Gold	4480	3.81	3.99	4.09	3.83	**4.40**	4.61
B612	4000	3.46	3.75	3.91	3.53	**4.22**	4.64
Flipkart	4480	2.33	2.46	**2.70**	2.45	2.60	3.80
Seven	4424	2.92	3.18	3.37	2.90	**3.51**	4.43
PhonePe	4000	2.78	2.97	3.13	2.78	**3.18**	3.98
Candy Crush Saga	4480	3.83	3.87	**4.04**	3.73	4.02	4.56
Pharmapedia Pakistan	4133	3.20	3.29	3.48	3.30	**3.68**	4.72
Beach Buggy Racing	4480	3.95	4.02	4.13	3.84	**4.17**	4.61

Bold terms represents the closet predicted rating of classifier to the actual ratings.

are closer to those of aggregate ratings than other classifiers. However, for "weather & clock widget," "Flipkart," and "Candy crush saga," GBM predicts a higher rating than other classifiers.

RF and AB yield the lowest predicted ratings for Google apps. RF is an ensemble technique that consists of joining multiple trees, thereby helping to overcome the limitations associated with single trees, that is, noise and outliers. RF is robust to noise and accurate. However, it is limited to a large collection of decision trees, owing to the counterintuitive nature of relationships within the input data [27]. The common assumption of RF is that sampling is representative. However, if the sampling is not fully representative, the prediction may be erroneous. AdaBoost is sensitive to noisy data and outliers. It converts weak learners into strong learners by applying weightings. By applying low weightings, AB can sometimes be trained on weak learners, resulting in overfitting.

GBM predicts higher ratings for two of the selected apps. In GBM, each new tree is fitted to a modified version of the original dataset. The Learning_ rate hyperparameter of GBM was set to 0.1 to avoid overfitting. Another reason for its performance is that it handles missing values more effectively than other machine learning models. This makes it a better predictor than other ensemble learning classifiers.

ET is similar to RF in terms of selecting a random subset of K features at each node for the tree split. However, unlike RF, it builds each tree from the complete learning sample and a cut-point is determined randomly to define a split, whereas RF selects the best cut-point based on the local sample. Thus, setting K to 1 results in a tree structure that is independent of the training set labels [39]. Similarly, ET produces piece-wise multilinear approximations rather than the piece-wise constant ones of RF. In ET, additional randomization smoothens the decision boundaries. The main reason why ET sometimes outperform RF is due to the additional randomness in the ensemble, which leads the base learners to make mistakes that are less correlated. It has been shown [31] that ET competes with RF and performs better in terms of accuracy.

XGB predicts less accurately compared to other classifiers because the classes assigned during the model training are imbalanced for the app considered. XGB is not effective when the division of the data is imbalanced. Additionally, tree-based bagging algorithms handle colinearity better than boosting algorithms. The stopping criteria for tree pruning and tree splitting are missing in boosting tree-based approaches [40]. If the number of features is large with a balanced class distribution, XGB may give the best performance by providing better average accuracy. Similarly, tree-based bagging algorithms also outperform boosting algorithms when dealing with categorical independent variables [41].

Considering the results obtained using different classifiers, GBM and ET visibly perform well and yield accurate predictions of app numeric ratings in all categories. These results are useful for evaluating App_ review and App_ rating. We can hence also conclude that numeric app ratings can be predicted based on user reviewers using ensemble machine learning algorithms.

The results also demonstrate the inconsistency of user numeric ratings and reviews. The analysis showed that no classifier can predict a higher numeric rating than the aggregate rating specified in the Google Play store. User-defined numeric ratings are typically higher than the review-based predictions. However, ensemble classifiers can be useful to highlight such contradictions and discrepancies.

5.3 | Performance of GBM and RF in predicting a given rating

A further analysis of RF and GBM performance was conducted separately by controlling the hyperparameters. First, RF was analyzed with TF/IDF and TF. The performance using TF/IDF was tested using uni-, bi-, and trigrams. Table 10 shows the accuracy of RF with the discussed VSM. The accuracy was determined for different values of the RF tuning parameter *n_estimator*. Experimental results show that RF is more accurate when the TF/IDF unigram is used for prediction. This is because user reviews are short and contain words like "Good," "great app," etc. This approach is suitable for the given dataset, which contains informal language (eg, with "please" abbreviated as "pls" or "plse").

The results demonstrate that the prediction accuracy of RF (ie, 72% when used with TF/IDF (unigram)) is closer to the original accuracy (75.28%). As previously explained, this accuracy represents the proportion of unbiased ratings out of the total user-specified ratings for a given app. The RF results confirm that approximately 25% of the total ratings are biased as the app ratings are consistent with sentiments expressed in the reviews.

Similarly, the classification accuracy of RF was analyzed using textual features and emoticons included in the reviews, and by including symbols and emoticons in the training phase. These inclusions reduced the classifier accuracy. This poor performance is presumably due to the relatively low number of reviews containing such emoticons. Since most reviews do not contain emoticons, adding them in the training phase favors improper training, which ultimately reduces the accuracy. The results listed in Table 11 were generated by adding emoticons in the training phase, causing a slight drop in accuracy.

The RF results were cross-checked with the boosting classifier. The GBM accuracy was evaluated by varying its learning rate between 0.05 and 1. The purpose of varying the learning rate is to avoid overfitting. GBM is quick to learn and overfit the training data. One effective way to slow down learning in GBM is to vary the learning rate. The corresponding GBM accuracies are shown in Table 12, indicating that GBM performs well with a learning rate between 0.5 and 0.75, achieving a prediction accuracy of 71%.

Experiments demonstrate that the prediction accuracy can be improved. For this purpose, each application category was trained separately using RF before making predictions. Table 13 shows the result obtained by training each category of Google apps using RF. The precision, recall, and F-score values were obtained by averaging the ratings from each individual class. Averages were calculated using the *Scikit learn* evaluation metrics library. The results demonstrate that training each category separately to make a prediction yields a higher accuracy than when using all categories combined.

The accuracy of RF and GBM reflects the discrepancies between the user-specified numeric rating and reviews. The numeric rating is approximately 20% higher than the outcome of ensemble classifiers.

TABLE 10 Accuracy of RF with different VSM techniques

VSM technique	Classification algorithm	Accuracy
TF/IDF	Random Forest	72%
TF/IDF(Bigram)	Random Forest	69%
TF/IDF(Trigram)	Random Forest	67%
TF	Random Forest	71%

TABLE 11 Comparison of RF accuracy with and without emoticons

VSM technique	Accuracy without Emoticons	Accuracy with Emoticons
TF/IDF	72%	71.67%
TF/IDF(Bigram)	69%	68.12%
TF/IDF(Trigram)	67%	67.50%
TF	71%	69.94%

TABLE 12 Accuracy of GBM for different learning rates

Learning rate	Classification algorithm	Accuracy
0.05	GBM	69%
0.1	GBM	70%
0.25	GBM	70%
0.5	GBM	71%
0.75	GBM	71%
1.0	GBM	70%

TABLE 13 Accuracy measures for the different app categories, obtained using RF

App category	Number Of reviews	Accuracy	Precision	Recall	F1-Score
Sports	32 280	64%	58%	64%	57%
Communication	30 000	63%	55%	63%	54%
Action	44 141	70%	63%	71%	63%
Arcade	40 751	74%	67%	74%	67%
Video players & editors	20 781	68%	60%	69%	61%
Weather	27 324	66%	58%	67%	59%
Card	36 520	70%	62%	70%	61%
Photography	41 440	72%	66%	73%	65%
Shopping	47 840	69%	65%	70%	63%
Health & fitness	34 415	81%	70%	81%	73%
Finance	28 220	63%	56%	64%	56%
Casual	52 560	75%	66%	75%	67%
Medical	24 002	77%	70%	77%	70%
Racing	42 384	74%	67%	74%	67%

6 | CONCLUSIONS AND FUTURE WORK

User reviews are limited to identifying polarity and subjectivity. However, the large increase in review-based data implies a need to focus also on performing predictions. This process is challenging yet fruitful, as user reviews are qualitative while ratings are essentially quantitative. The numeric scoring of apps in the Google App store may be biased and overrated because higher ratings given by users potentially attract several new users disproportionately. This study therefore investigated the use of ensemble classifiers to predict numeric ratings for Google Play store apps based on the user reviews for those apps. Several ensemble classifiers were investigated to evaluate their performance on the reviews scraped from the Google App store. TF/IDF features with unigrams, bigrams, and trigrams were utilized with the selected classifiers. The ground truth was calculated using a technique based on TextBlob analysis, which identifies the reviews showing a discrepancy with the user-awarded rating. Subsequently, it was used to evaluate the performance of the selected classifiers. TextBlob analysis showed that 24.72% of the user-defined app ratings are biased. Results demonstrate that tree-based bagging ensemble classifiers perform much better than boosting-based classifiers on account of their support for nonlinearity, colinearity, and tolerance to data noise. The analysis also reveals that the user reviews are inconsistent with user numeric ratings, and that numeric ratings are higher than user reviews might suggest. Future work includes the implementation of the deep learning technique to predict numeric rating.

CONFLICT OF INTEREST

The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

ORCID

Muhammad Umer https://orcid.org/0000-0002-6015-9326
Saleem Ullah https://orcid.org/0000-0003-3747-1263

REFERENCES

1. Statista, Number of available application in the Google Play store from December 2009 to March 2019, https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/, Online: accessed 22 May 2019.
2. Statistaa, Number of mobile app downloads worldwide in 2017, 2018 and 2020 (in billions), https://www.statista.com/statistics/271644/worldwide-free-and-paid-mobile-app-store-downloads/, Online: accessed 22 May 2019.
3. J. Horrigan, *Online shopping, pew internet and american life project*, Washington, DC, 2018, http://www.pewinternet.org/Reports/2008/Online-Shopping/01-Summary-of-Findings.aspx Online: accessed 8 Aug. 2014.
4. D. Pagano and W. Maalej, *User feedback in the appstore: An empirical study*, in Proc. IEEE Int. Requirements Eng. Conf. (Rio de Janeiro, Brazil), July 2013, pp. 125–134.
5. T. Chumwatana, *Using sentiment analysis technique for analyzing Thai customer satisfaction from social media*, 2015.
6. T. Thiviya et al., *Mobile apps' feature extraction based on user reviews using machine learning*, 2019.
7. H. Hanyang et al., *Studying the consistency of star ratings and reviews of popular free hybrid android and ios apps*, Empirical Softw. Eng. **24** (2019), no. 7, 7–32.

8. N. Kumari and S. Narayan Singh, *Sentiment analysis on e-commerce application by using opinion mining*, in Proc. Int. Conf.- Cloud Syst. Big Data Eng. (Noida, India), Jan. 2016, pp. 320–325.
9. R. M. Duwairi and I. Qarqaz, *Arabic sentiment analysis using supervised classification*, in Proc. Int. Conf. Future Internet Things Cloud (Barcelona, Spain), Aug. 2014, pp. 579–583.
10. H. S. Le, T. V. Le, and T. V. Pham, *Aspect analysis for opinion mining of vietnamese text*, in Proc. Int. Conf. Adv. Comput. Applicat. (Ho Chi Minh, Vietnam), Nov. 2015, pp. 118–123.
11. H. Wang, L. Yue, and C. Zhai, *Latent aspect rating analysis on review text data: A rating regression approach*, in Proc. ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining (Washington, D.C., USA), July 2010, pp. 783–792.
12. K. Dave, S. Lawrence, and D. M. Pennock, *Mining the peanut gallery: Opinion extraction and semantic classification of product reviews*, in Proc. Int. Conf. World Wide Web (New York, USA), 2003, pp. 519–528.
13. B. Pang, L. Lee, S. Vaithyanathan, *Thumbs up?: Sentiment classification using machine learning techniques*, in Proc. ACL-02 Conf. Empirical Methods Natural Language Process. (Stroudsbrug, PA, USA), 2002, pp. 79–86.
14. C. Cardie et al., *Combining low-level and summary representations of opinions for multi-perspective question answering*, New directions in question answering, 2003, pp. 20–27.
15. H. Takamura, T. Inui, and M. Okumura, *Extracting semantic orientations of words using spin model*, in Proc. Annu. Meeting Association Comput. Linguistics (Ann Arbor, MI, USA), 2005, pp. 133–140.
16. A. Buche, D. Chandak, and A. Zadgaonkar, *Opinion mining and analysis: A survey*, arXiv preprint arXiv:1307.3336, 2013.
17. 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, 2019.
18. F. Sarro et al., *Customer rating reactions can be predicted purely using app features*, in Proc. IEEE Int. Requirements Eng. Conf. (Banaf, Canada), Aug. 2018, pp. 76–87.
19. S. Aslam and I. Ashraf, *Data mining algorithms and their applications in education data mining*, Int. J. Adv. Res. Computer Sci. Manag. Studies **2** (2014), no. 7, 50–56.
20. D. Martens and T. Johann, *On the emotion of users in app reviews*, in Proc. IEEE/ACM Int. Workshop Emotion Awareness Softw. Eng. (Buenos Aires, Argentina), May 2017, pp. 8–14.
21. G. Hackeling, *Mastering machine learning with scikit-learn*, Packt Publishing Ltd, 2017.
22. Scikit learn, *Scikit-learn classification and regression models*, http://scikitlearn.org/stable/supervised_learning.html#supervised-learning/, Online: accessed 10 Apr. 2019
23. O. Araque et al., *Enhancing deep learning sentiment analysis with ensemble techniques in social applications*, Expert Syst. Appl. **77** (2017), 236–246.
24. J. Hartmann et al., *Comparing automated text classification methods*, Int. J. Res. Mark. **36** (2019), 20–38.
25. O. Aziz et al., *A comparison of accuracy of fall detection algorithms (threshold-based vs. machine learning) using waistmounted tri-axial accelerometer signals from a comprehensive set of falls and non-fall trials*, Med. Biol. Eng. Comput. **55** (2017), no. 1, 45–55.
26. Z. Hailong, G. Wenyan, and J. Bo, *Machine learning and lexicon based methods for sentiment classification: A survey*, in Proc. Web Inf. Syst. Applicat. Conf. (Tianjin, China), Sept. 2014, pp. 262–265.
27. L. Breiman, *Random forests*, Mach. Learn. **45** (2001), no. 1, 5–32.
28. R. E. Schapire and Y. Singer, *Improved boosting algorithms using confidence-rated predictions*, Mach. Learn. **37** (1999), no. 3, 297–336.
29. A. Natekin and A. Knoll, *Gradient boosting machines, a tutorial*, Frontiers Neurorobotics **7** (2013), 21.
30. T. Chen and C. Guestrin, *Xgboost: A scalable tree boosting system*, in Proc. ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining (San Francisco, CA, USA), Aug. 2016, pp. 785–794.
31. P. Geurts, D. Ernst, and L. Wehenkel, *Extremely randomized trees*, Mach. Learn. **63** (2006), no. 1, 3–42.
32. R. Feldman and J. Sanger, *The text mining handbook: Advanced approaches in analyzing unstructured data*, Cambridge University Press, 2007.
33. B. Sriram et al., *Short text classification in twitter to improve information filtering*, in Proc. Int. ACM SIGIR Conf. Res. Development Inf. Retrieval (Geneva, Switzerland), July 2010, pp. 841–842.
34. Scikit learn, *Scikit-learn feature extraction with countvectorizer*, https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.Count/, Online: accessed 5 Apr. 2019
35. Scikit learn, *Scikit-learn feature extraction with tf/idf*, https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.Tfidf/, Online: accessed 5 Apr. 2019
36. J. Han, J. Pei, and M. Kamber, *Data mining: Concepts and techniques*, Elsevier, 2011.
37. I. Ashraf, S. Hur, and Y. Park, *Blocate: A building identification scheme in gps denied environments using smartphone sensors*, Sensors **18** (2018), no. 11, 3862.
38. S. Loria, *textblob documentation*, Release 0.15 2 (2018).
39. P. Geurts and G. Louppe, *Learning to rank with extremely randomized trees*, JMLR: Workshop Conf. Proc. **14** (2011) 49–61.
40. X. Z. Fern and C. E. Brodley, *Boosting lazy decision trees*, In Proc. Int. Conf. Mach. Learn., 2003, pp. 178–185.
41. L. Breiman, *Randomizing outputs to increase prediction accuracy*, Mach. Learn. **40** (2000), no. 3, 229–242.

AUTHOR BIOGRAPHIES

Muhammad Umer received his BS degree at the Department of Computer Science, Khwaja Fareed University of Engineering & IT(KFUEIT), Pakistan (October 2014 to October 2018). Since November 2018, he is enrolled in the Master of Computer Science(KFUEIT) program. He is also serving as a Research Assistant at the Fareed Computing & Research Center, KFUEIT, Pakistan. His recent research interests are related to data mining, mainly machine learning and deep-learning-based IoT, text mining, and computer vision tasks.

Imran Ashraf received his PhD in Information & Communication Engineering from Yeungnam University, South Korea in 2018 and MS degree in computer science from the Blekinge Institute of Technology, Karlskrona, Sweden, in 2010. He is currently working as an Assistant Professor at the Information and Communication Engineering Department, Yeungnam University, Gyeongsan, South Korea. His research areas include indoor positioning and localization, indoor location-based services in wireless communication, and data mining & data analytics.

Arif Mehmood received his PhD degree at the Department of Information & Communication Engineering, Yeungnam University, Korea (February 2014 to November 2017). He worked as an Assistant Professor at the Department of Computer Science, KFUEIT, Pakistan from November 2017 to December 2019. Currently, he is at the Islamia University of Bahawalpur, Pakistan. His recent research interests are related to data mining, with main focus on AI and deep learning-based text mining, and data science management technologies.

Saleem Ullah was born in Ahmed Pur East, Pakistan in 1983. He received his BSc and MIT degrees in Computer Science from Islamia University Bahawalpur and Bahauddin Zakariya University (Multan) in 2003 and 2005, respectively. From 2006 to 2009, he worked as a Network/IT Administrator in different companies. He obtained his doctoral degree from Chongqing University, China in 2012. From August 2012 to February 2016, he worked as an Assistant Professor at the Islamia University Bahawalpur, Pakistan. Currently, he is working as an Associate Professor in Khwaja Fareed University of Engineering & Information Technology, Rahim Yar Khan since February 2016. He has nearly 13 years of industrial experience in the field of IT. He is an active researcher in the field of ad hoc networks, congestion control, and security.

Gyu Sang Choi received his PhD at the Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA, USA in 2005. He was a research staff member at the Samsung Advanced Institute of Technology (SAIT) for Samsung Electronics from 2006 to 2009. Since 2009, he has been a faculty member in the Department of Information & Communication, Yeungnam University, Korea. His research areas include nonvolatile memory and storage systems.