IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 26, 2019, accepted February 1, 2020, date of publication February 10, 2020, date of current version February 18, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2972632

Classification of Shopify App User Reviews Using Novel Multi Text Features

FURQAN RUSTAM[1], ARIF MEHMOOD[4], MUHAMMAD AHMAD[2,3], SALEEM ULLAH[1], DOST MUHAMMAD KHAN[4], AND GYU SANG CHOI[5]

[1]Department of Computer Science, Khwaja Fareed University of Engineering and Information Technology, Rahim Yar Khan 64200, Pakistan
[2]Department of Computer Engineering, Khwaja Fareed University of Engineering and Information Technology, Rahim Yar Khan 64200, Pakistan
[3]Dipartimento di Matematica e Informatica—MIFT, University of Messina, 98121 Messina, Italy
[4]Department of Computer Science and IT, The Islamia University of Bahawalpur, Bahawalpur 63100, Pakistan
[5]Department of Information and Communication Engineering, Yeungnam University, Gyeongsan 38541, South Korea

Corresponding authors: Arif Mehmood (arifnhmp@gmail.com) and Gyu Sang Choi (castchoi@ynu.ac.kr)

This work was supported in part by the Ministry of Trade, Industry and Energy (MOTIE, South Korea) through the Industrial Technology Innovation Program under Grant 10063130, and in part by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MSIT) under Grant NRF-2019R1A2C1006159.

ABSTRACT App stores usually allow users to give reviews and ratings that are used by developers to resolve issues and make plans for their apps. In this way, these app stores collect large amounts of data for analysis. However, there are several challenges that must first be addressed, related to redundancy and the volume of data, by using machine learning. This study performs experiments on a dataset that contains reviews for Shopify apps. To overcome the aforementioned limitations, we first categorize user reviews into two groups, i.e., happy and unhappy, and then perform preprocessing on the reviews to clean the data. At a later stage, several feature engineering techniques, such as bag-of-words, term frequency-inverse document frequency (TF-IDF), and chi-square (Chi2), are used singly and in combination to preserve meaningful information. Finally, the random forest, AdaBoost classifier, and logistic regression models are used to classify the reviews as happy or unhappy. The performance of our proposed pipeline was evaluated using average accuracy, precision, recall, and f_1 score. The experiments reveal that a combination of features can improve machine learning models performance and in this study, logistic regression outperforms the others and achieves an 83% true acceptance rate when combined with TF-IDF and Chi2.

INDEX TERMS Feature engineering, feature extraction, feature selection, machine learning, review classification, text mining.

I. INTRODUCTION

Manufacturers always want to know the success rate of their products/apps, and for that, they usually request users to provide feedback that is later used to analyze the impact and quality of their products [1], [2]. However, it is critical to analyze such feedback due to the volume and redundancy. Therefore, this work investigates an efficient way to analyze such feedback and solve the problems related to the classification of Shopify app reviews. Though there are many techniques that have already been proposed, such as [3], which performed topic modeling to find high-level and meaningful features using latent Dirichlet allocation to filter the irrelevant reviews and achieved 59% precision and 51% recall rates.

The work [4] built a mobile app review analyzer that automatically extracts user requests or suggestions from reviews. This app analyzer works on the basis of linguistic rules to extract requests from online reviews. The work [5] presented some probabilistic techniques for classifying app reviews. They classified these reviews into four categories: ratings, bug reports, feature requests, and user experiences. They used multiple binary classifiers to classify reviews and achieve acceptable results. The work [6] used different machine learning algorithms to solve app review classification problems. They performed a comparative analysis of the results of

The associate editor coordinating the review of this manuscript and approving it for publication was Yongming Li.

machine learning algorithms, such as naïve Bayes (NB), random forest (RF), decision tree (DT), support vector machine (SVM), and logistic regression (LR). As per previous work, the researchers tried to solve different problems during app review analysis. This study solved the classification problems of Shopify app reviews on the basis of the ratings given by app users and performed a comparative analysis between tree-based ensemble and linear models.

More formally, machine learning algorithms take the users' reviews as input and then perform analysis on these reviews to predict whether the users are happy or not. This work does not investigate other types of information, such as user properties, app names, and app descriptions. We intentionally limited the inputs (i.e., user reviews and rating scores) to keep the problem definition simple. We used a dataset obtained from Kaggle, on which preprocessing using the natural language toolkit (NLTK) [7] has been done to clean the reviews. Further preprocessing steps for cleaning the reviews included tokenization, punctuation removal, lower-case conversion, removal of numeric values, and stopword removal. Finally, the stemming technique was used to get the root-form of each feature in the reviews. Rating scores from users (1 *to* 5) were used to create two classes, i.e., happy and unhappy, where the users who gave rating scores of 3 or above were assigned as happy, and the rest are unhappy. More details regarding this phenomenon can be found in section IV(B).

After preprocessing, two different text feature extraction techniques, bag-of-words (BoW) and term frequency-inverse document frequency (TF-IDF), were deployed to extract the high-level features. Later, the chi-square (Chi2) feature selection technique was used to select the most important and less redundant features to fit several machine learning models.

Prior to fitting the model, the data must be split into two parts, i.e., training and testing, in a 70% to 30% ratio. Finally, several machine learning classifiers, i.e., AdaBoost classifier (AC), LR, and RF, were used to classify the reviews as happy or unhappy. RF and AC are both tree-based ensemble models, while LR is a statistical method to solve classification problems [8]. For this study, accuracy, precision, recall, and score metrics are used as performance evaluation metrics. The key points of this study are as follows:

- Categorization of happy and unhappy users on the basis of reviews and ratings
- Preprocessing techniques to clean text reviews for efficient learning of models, i.e., stemming, stopwords removal techniques, convert to lower case, punctuation, and numeric value removal technique
- Feature Engineering techniques, i.e., BoW, TF-IDF, Chi2
- Machine learning models, i.e., RF, AC, and LR
- Comparative analysis of the performance of learning models with respect to feature engineering techniques

The rest of this paper is organized as follows: Section II presents related work. Section III describes the material and methods used in this study. Section IV contains the proposed methodology, and Section V contains the results and discussion. Finally, section VI concludes the paper with possible directions for future research.

II. RELATED WORK

As mentioned above, data classification is an area explored by many data scientists. Researchers have done much work in the text classification domain, using different approaches and introducing some new techniques in this field. In this section, we discuss previous work on app review classification and analysis.

The study [9] works on app review classification using ensemble algorithms and techniques. The dataset used in the study was previously examined in [3], the dataset contains reviews from Apple's app store and the Google Play app store. In the study [9], the authors used NB, SVM, LR, and neural network (NN) in various combinations for classification. They built three ensemble algorithms A, B, and C. In ensemble A, four classifiers, NB, SVM, LR, and NN, were grouped for final prediction; in ensemble B, three classifiers, SVM, LR, and NN, were grouped, and in ensemble C, the two classifiers NB and SVM were grouped. The best performers from these individual and ensembles algorithms were LR and NN. This study also used ensemble models, such as RF and AC, which work with numbers of base learners (decision trees) to make final predictions.

In another research [4], text analysis was performed for mobile app feature requests. They designed MARA (mobile app review analyzer), a prototype for automatic retrieval of mobile app feature requests from online reviews. MARA takes review content as input for feature request mining. The feature request mining algorithm uses a set of linguistic rules, which are defined for supporting the identification of sentences that indicate such requests. The linear discriminant analyzer model was used to identify topics that can be associated with these requests in user reviews. They used true positive (TP), false positive (FP), true negative (TN), false negative (FN), precision, recall, and Matthews correlation coefficient as evaluation metrics to check the accuracy of the algorithm.

Researchers perform analysis on app reviews to facilitate app developers in finding out whether their customers are happy are not, which is also a goal of this study. In study [10], researchers tried to help mobile app developers by performing analysis on user reviews to categorize information that is important for app maintenance and evolution. For classification purposes, they deduced a taxonomy of user review categories that are relevant to app maintenance. The authors merged three techniques, natural language processing, text analysis, and sentiment analysis.

By merging these techniques, they achieved desirable results in terms of precision and recall (Precision Score 74% and Recall Score 73%). They also applied these techniques individually to classify user reviews. In another study [11], the authors tried to extract the values of comparison scores

of sentiment reviews using different feature extraction techniques, such as word2vec, word2doc, and TF-IDF, with SVM, NB, and decision tree algorithms. In study [11], the authors used grid search algorithms for parameter optimization of machine learning algorithms and feature extraction methods.

LR performs significantly better in the case of classification, but LR is usually preferred by researchers when there is a binary classification problem. Study [12] used LR for tweet classification with different feature engineering techniques and achieved acceptable results. Similarly, the other two ensemble models that were used in this study, RF and AC, are also used in many fields of data mining. These tree-based ensemble models perform well on text data and categorical data. Study [13] used RF in its research to predict a chemical compound's quantitative or categorical biological activity based on a quantitative description of the compound's molecular structure. They trained RF using six cheminformatics datasets. Their analysis showed that RF is a powerful tool to give good performance and accurate models. RF is an ensemble model that creates trees using bootstrap samples of training data.

III. MATERIALS AND METHODS

A. DATA DESCRIPTION

This study used the Shopify app store dataset, which was obtained from Kaggle, a well-known source for benchmark datasets. The dataset contains 287467 examples and seven variables about Shopify apps. This study used two of the seven variables (reviews and ratings) for its experiments. The dataset contains reviews for different categories of apps, such as store design, finance, orders and shipping, marketing, reporting, trust and security, finding and adding products, inventory management, reporting, productivity, and places to sell. In the reviews, users express their problems and issues related to apps and also give ratings to apps from 1 to 5.

Table 1 shows the variables contained in the Shopify app store dataset, and Table 2 shows examples from the dataset used in this study.

The dataset contains rating scores from 1 to 5 corresponding to each review, but in the dataset, all rating score ratios are not equal, as shown in Table 3. Rating score 3 has the lowest number of examples in the dataset, 2552, and rating score 1 contains 246712 examples, which makes the dataset highly imbalanced. To solve this problem, we extracted 2552 examples from each rating score to balance the dataset. For the experiment, a total of 12760 examples were used in this study, which is 2552 examples from each rating score.

TABLE 1. Description of dataset variables.

Variables	Description
app_url	App url, can be joined to apps
url	URL of web page where review was taken from (with get parameters)
author	The title of author
body (reviews)	The body of review
rating	The number of stars that author assigned to the app
helpful_count	Number of times the review was considered as helpful
poste_at	Date when the review was written

TABLE 2. Sample of data from dataset.

Body	Rating
Great app	5
Update - the bugs reported 5 months ago still arent fixed	2
SUPER Easy to install!! Thank you!	5
Really terrible app so far.	1
It's ok. App design isn't that great. Product works though	3

TABLE 3. Number of samples corresponding to each rating score.

Rating Score	Total number of examples in dataset	Number of examples taken for experiemnts
1	246712	2552
2	23507	2552
3	5783	2552
4	2552	2552
5	8913	2552

B. DATASET VISUALIZATION

This section illustrates the make-up of the dataset graphically.

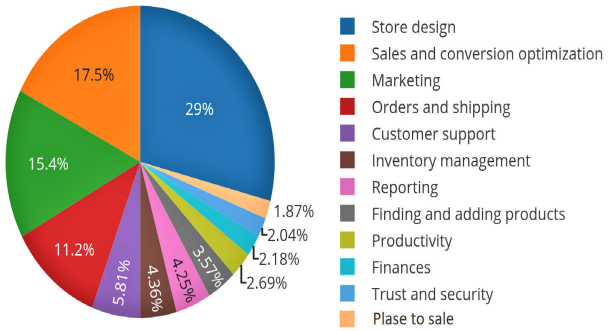


FIGURE 1. Apps per category.

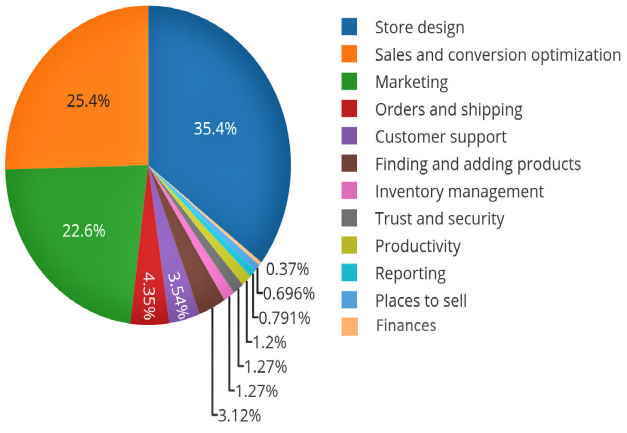


FIGURE 2. Reviews per category.

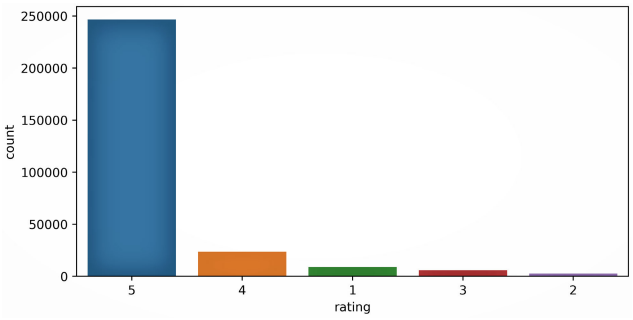


FIGURE 3. Per rating score corresponding to user reviews on apps.

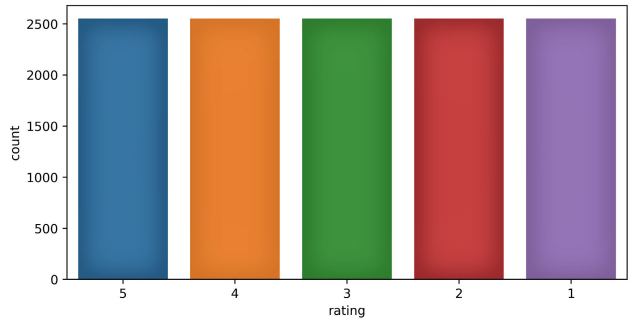


FIGURE 4. After extracting 2552 examples from each rating score.

C. FEATURE ENGINEERING METHODS

Feature engineering is a process for finding meaningful features from data for the efficient training of machine learning algorithms or, in other words, the creation of features derived from original features [14]. The study [15] concludes that feature engineering can boost the performance of machine learning algorithms. ''Garbage in garbage out'' is a common saying in machine learning. According to this idea, senseless data produces senseless output. On the other hand, data that are more informational can produce desirable results. Therefore, feature engineering can extract meaningful features from raw data, which helps to increase the consistency and accuracy of learning algorithms. In this study, we used three feature engineering methods: BoW, TF-IDF, and Chi2.

1) BAG-OF-WORDS

BoW is a method of extracting features from text data, and it is very easy to understand and implement. BoW is very useful in problems such as language modeling and text classification. In this method, we use CountVectorizer to extract features. CountVectorizer works on term frequency, i.e., counting the occurrences of tokens and building a sparse matrix of tokens [16]. BoW is a collection of words and features, where each feature is assigned a value that represents the occurrences of that feature [17].

2) TF-IDF

TF-IDF is a feature extraction method used to extract features from data. TF-IDF is most widely used in text analysis and music information retrieval [18]. TF-IDF assigns a weight to each term in a document based on its term frequency (TF) and inverse document frequency (IDF) [12], [19]. The terms with higher weight scores are considered to be more important [20]. TF-IDF computes weight of each term by using formula as mention in equation 1:

$$W_{i,j} = TF_{i,j}(\frac{N}{D_{f,t}}) \tag{1}$$

Here, $TF_{i,j}$ is the number of occurrences of term t in document d, $D_{f,t}$ is the number of documents containing the term t, and N is the total number of documents in the corpus.

3) CHI2

Chi2 is the most common feature selection method, and it is mostly used on text data [21]. In feature selection, we use it to check whether the occurrence of a specific term and the occurrence of a specific class are independent. More formally, forgiven a document D, we estimate the following quantity for each term and rank them by their score. Chi2 finds this score using equation 2:

$$X^2(D,t,c) = \sum_{e_t \in \{0,1\}} \sum_{e_c \in \{0,1\}} \frac{(N_{e_t e_c} - E_{e_t e_c})^2}{E_{e_t e_c}} \tag{2}$$

where

- N is the observed frequency and E the expected frequency
- e_t takes the value 1 if the document contains term t and 0 otherwise
- e_c takes the value 1 if the document is in class c and 0 otherwise

For each feature (term), a corresponding high Chi2 score indicates that the null hypothesis H0 of independence (meaning the document class has no influence over the term's frequency) should be rejected, and the occurrence of the term and class are dependent. In this case, we should select the feature for the text classification.

D. SUPERVISED MACHINE LEARNING METHODS

In this section, we discuss the machine learning algorithms; we describe the implementation details of machine learning algorithms and their hyperparameters. Scikit-learn library and NLTK were used for the implementation of machine learning algorithms [7], [22].

All three supervised machine learning algorithms mentioned below were deployed in Python using the scikit module. Supervised machine learning algorithms are commonly used to solve classification and regression problems [23]. RF and AC are tree-based algorithms, while LR is a linear model. This study used three different machine-learning algorithms to solve the classification problem. The implementation details of the algorithms with their hyperparameters are shown in Table 4.

TABLE 4. Machine learning algorithms and their hyperparameters.

Learning Algorithms	Hyperparameters
LR	solver="liblinear" multi_class="ovr" C=3.0
RF	n_estimator=300 max_depth=60 random_state=2
AC	n_estimator=300 random_state=2

1) RANDOM FOREST (RF)

RF is a tree-based ensemble model that produces highly accurate predictions by combining many weak learners (decision trees) [13]. This model uses the bagging technique to train a number of decision trees using different bootstrap samples [24]. In RF, a bootstrap sample is obtained by subsampling the training dataset with replacement, where the size of a sample is the same as that of the training dataset [25]. RF and other classifiers that use decision trees in their prediction procedure apply the same techniques to construct decision trees, and a major challenge in constructing them is the identification of the attribute for the root node at each level.

This process is known as attribute selection [26]. In ensemble classification, several classifiers are trained, and their results are combined through a voting process. In the past, many researchers have proposed ensemble methods [27]–[29]. The most popular ensemble methods are bagging [25] and boosting [30], [31]. Bagging (or bootstrap aggregating) is a method in which many classifiers train on bootstrapped samples, which has been shown to reduce the variance of the classification. RF can be defined as in equation 3 and 4:

$$p = mode\{T_1(y), T_2(y), \ldots, T_m(y)\} \tag{3}$$

$$p = mode\{\sum_{m=1}^{m} T_m(y)\} \tag{4}$$

Here p is the final prediction by majority voting of decision trees, while $T_1(y)$, $T_2(y)$, $T_3(y)$, and $T_m(y)$ are the number of decision trees participating in the prediction procedure. For a more detailed discussion about the RF algorithm, see [24].

RF was implemented with 300 weak learners to get high accuracy, which is the reason we set the n_estimator value equal to 300. Parameter n_estimator defines the number of trees that are contributing to prediction. In the experiment, RF trained 300 decision trees with bootstrap samples, and the final prediction was made by the voting between all decision trees predictions [32]. The second parameter used in RF was ''max_depth'' with a value of 60. This max_depth parameter was used to set the maximum depth level of each decision tree. This max_depth parameter reduces complexity in the decision tree by setting the depth level and reduces the chances of overfitting of the decision tree [25]. Another parameter used for the RF algorithm was ''random_state.'' This parameter was used for the randomness of the samples during the training of the model. By using these two hyperparameters, we achieved good results with RF in this study.

2) ADABOOST CLASSIFIER (AC)

This is an ensemble learning model that uses a boosting method for training weak learners (decision trees). Adaboost is an acronym for adaptive boosting. AC is very popular and possibly the most historically significant as it was the first algorithm that could adapt to weak learners [33]. AC algorithm combines numbers of ''weak learners'' and trains them recursively on duplicates of the original dataset, while all weak learners focus on the difficult data points or outliers [31]. It is a meta-model that takes N copies of weak learners and trains them on the same feature set but with different weights assigned to them. The study [34], eulogize the performance of AC with mathematical ground truth in their research. In many classification experiments, the AC has been shown to outperform the other machine learning algorithms [35]–[37]. For a detailed discussion of AC, see [38] section 2.

This study implemented the AC algorithm with different hyperparameters (see Table 4) and tuned these parameters to get high accuracy. Parameter ''n_estimator'' was used with a value of 300, which means that the AC algorithm combined 300 week learners to make some predictions. The difference between RF and AC ensemble learning is that RF uses the bagging method, while AC uses the booting method, and it is exactly the weighted combination of N weak learners. Another parameter that we use is ''random_state,'' this parameter defines the randomness of the samples during the training of the model. AC classification equation can be represented as in equation 5:

$$f(v) = sign(\sum_{t=1}^{T} \theta_t f_t(v)) \tag{5}$$

where

- f_t is the n^{th} weak learner
- θ_t is the weight of n^{th} weak learner

AC model identifies the outlier using high weight data points, and the gradient boosting algorithm performs the same task using gradients in the loss function [31].

3) LOGISTIC REGRESSION (LR)

LR is a statistical method for analyzing data in which there are one or more variables used to find the outcome. LR is the regression model that was used to estimate the probability of class members, so it is the best learning model to use when the target variable is categorical. LR processes the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function. A logistic function or logistic curve is a common "S" shaped, or sigmoid curve, as in equation 6:

$$f(x) = \frac{L}{1 + e^{-m(v - v_o)}} \tag{6}$$

where,

- e is the natural algorithm base (also known as Euler Number).
- v_o is the x-value of the sigmoid midpoint.
- L is the curve's maximum value.
- m is the steepness of the curve.

For values of v in the domain of real numbers from $-\infty$ to $+\infty$, the S-curve of logistic function will be obtained, with the graph of F approaching L as v approaches $+\infty$ and approaching zero as x approaches $-\infty$. This study used the liblinear algorithm for optimization because it works well on small datasets, whereas "sag" and "saga" are faster for large ones. The second parameter that was used in this study with AC is "multi_class," and we used it with the "ovr" value because it is good for binary classification. The third parameter is "C." This is the inverse regularization parameter that holds the strength modification of regularization by being inversely positioned to the Lambda regulator and reduces the chance of overfitting the model [39]. LR models were used in this study because LR is better for binary classification and also effective for categorizing text [3], [40].

IV. METHODOLOGY

In this section, we formulate the problem and the assumptions and then describe the method and details of the techniques we used to solve the user's classification problem.

A. PROBLEM STATEMENT

Assume that a company launches an app that helps its customers over the internet. Customers use this app and face some issues related to the app, which makes them uncomfortable. Customers want the company to solve these issues. For this, the company gives the option for its customers to give reviews about the services or about any issues related to the app. Many customers give their reviews about the company's products or services. Then the company performs an analysis of these reviews to find the good things and the bad things and tries to determine whether the customer is happy or not, which is very helpful for their business strategy. In other words, the problem is to predict whether the customer is satisfied or not. As described in section I, to solve this problem, we use text features of reviews and rating scores as input.

B. PROPOSED METHODOLOGY

This study uses different techniques to solve classification problems, as shown in Figure 5.

Figure 5 illustrates the steps for solving the user's classification problem. First, data goes through the preprocessing phase. As discussed in section III, that dataset has ratings from 1 to 5 that correspond to each review. The study places the ratings into two classes, happy and unhappy. Ratings equal to or greater than 3 are assigned to the happy class, and ratings less than 3 are assigned to the unhappy class, as shown in Table 5.

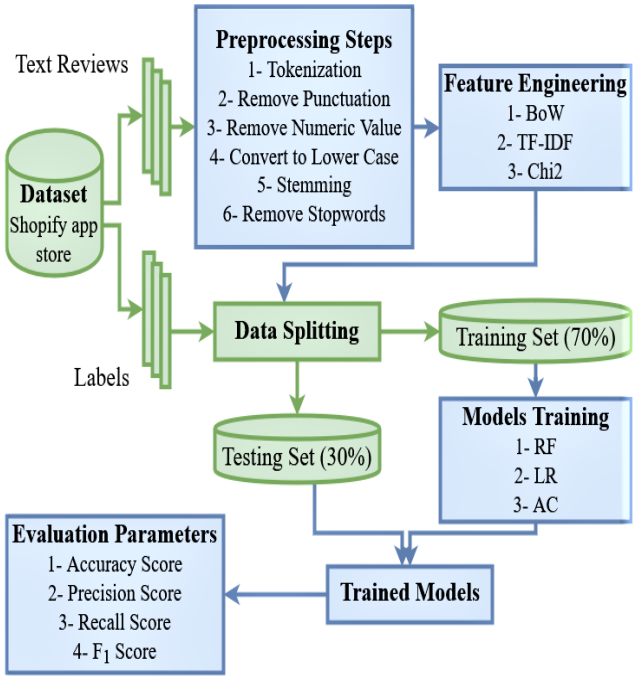


FIGURE 5. Methodology diagram: Green color represent the data flow while light blue color represents the techniques and methods.

In the next step of the experiment, review texts are cleaned by performing preprocessing steps, applying tokenization to reviews, and then numeric values are removed from reviews. Usually, numeric digits have no influence on the meaning of the text. In the next step of preprocessing, convert letters to lower case and remove punctuation from text reviews, because punctuation is not valuable for text analysis [12]. Sentences might be more readable because of punctuation, but it is difficult for a machine to differentiate punctuation from other characters. For that reason, punctuation was removed from the text during preprocessing. Then a stemming technique was applied to the reviews to get the root form of each word using the PorterStemmer library [41]. At the end of preprocessing, stopwords are removed from text reviews because they create confusion in text analysis. Table 6 shows a sample data extract from the prepared dataset, and Table 7 shows the results after preprocessing the sample data.

After preprocessing, we split the dataset into two subsets for training and testing. We divided the dataset in a ratio of 70% to 30%. For training, use 70% of the data, and for testing, use 30%. Feature engineering techniques (see section III(C)) were performed on both training and testing

TABLE 5. Number of samples corresponding to each rating score.

Reviews sample	Rating	Target Class
Super application gratuite ! Great and free application!	5	Happy
Update - the bugs reported 5 months ago still arent fixed	2	Unhappy
Brilliant service!	5	Happy
Really terrible app so far.	1	Unhappy
It's ok. App design isn't that great. Product works though	3	Happy

TABLE 6. Prepared dataset sample after conversion of rating into target classes.

Reviews sample	Target Class
Super application gratuite ! Great and free application!	Happy
Update - the bugs reported 5 months ago still arent fixed	Unhappy
Brilliant service!	Happy
Really terrible app so far.	Unhappy
It's ok. App design isn't that great. Product works though	Happy

TABLE 7. Preprocessing of sample reviews.

Before Preprocessing	After Preprocessing
Super application gratuite ! Great and free application!	super application gratuite great free application
Update - the bugs reported 5 months ago still arent fixed	update bug report month ago still arent fix
Brilliant service!	brilliant service
Really terrible app so far.	really terrible app far
It's ok. App design isn't that great. Product works though	ok app design isnt great product work though

sets to select and extract and important features from text reviews. The BoW and TF-IDF techniques, which are commonly used in text classification where the frequency of each word is used as a feature for training a classifier [42], were used for feature extraction. Three reviews were used as sample data (Table 8) to apply BoW and TF-IDF techniques, and the results are shown in Table 9 and 10 below.

TABLE 8. Sample of reviews.

No.	Sample of reviews without preprocessing	After preprocessing
1	Brilliant service!	brilliant service
2	Really terrible app so far.	really terrible app far
3	'good companies'	good company

TABLE 9. Results of BoW technique on preprocessed sample data.

No.	brilliant	service	really	terrible	app	far	good	company
1	1	1	0	0	0	0	0	0
2	0	0	1	1	1	1	0	0
3	0	0	0	0	0	0	1	1

TABLE 10. Results of TF-IDF technique on preprocessed sample data.

No.	brilliant	service	really	terrible	app	far	good	company
1	0.0000	0.7071	0.0000	0.0000	0.0000	0.0000	0.7071	0.0000
2	0.5000	0.0000	0.0000	0.5000	0.0000	0.5000	0.0000	0.5000
3	0.0000	0.0000	0.7071	0.0000	0.7071	0.0000	0.0000	0.0000

Table 9 shows the frequency of each feature in the sample data. Table 10 shows the weight of each feature in the sample data. The Chi2 feature selection technique was applied to the results of BoW and TF-IDF to select important features from the data. After feature engineering, machine learning models train using important features that were extracted by feature engineering techniques. Machine learning models tune with different hyperparameters as mentioned in Table 4. After model training, test data was passed to the trained models to evaluate the performance of the learning model.

C. EVALUATION CRITERIA

After all these steps, we come to the prediction phase. This study used several evaluation metrics, which are accuracy, f_1 score, recall, and precision. These evaluation parameters are used to evaluate machine learning models [43]. This study also used confusion matrices to evaluate the performance algorithms; a confusion matrix is a table that is mostly used to describe the performance of a classifier on test data. It is also known as an error matrix that allows visualization of the performance of an algorithm.

1) ACCURACY

The accuracy score used to measure prediction correctness for labels or target classes. This score's highest value is 1, and the lowest value is 0.

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions} \tag{7}$$

For binary classification, accuracy can also be calculated in terms of positives and negatives as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

where

- **True Positives (TP):** the model predicted *happy* (the user is happy), and the actual value is also *happy*.
- **True Negatives (TN):** the model predicted *unhappy* (the user is not happy), and the actual value is also *unhappy*.
- **False Positives (FP):** the model predicted *unhappy*, but the actual value is *happy*. (Also known as a "Type I error.").
- **False Negatives (FN):** the model predicted *happy*, but the actual value is *unhappy*. (Also known as a "Type II error.").

2) RECALL

Recall is the completeness of our classifiers. Recall is the number of true positives divided by the number of true positives plus the number of false negatives. The highest value is 1, and the lowest value is 0.

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

3) PRECISION

Precision is the exactness of our classifiers. Precision is the number of true positives divided by the number of true positives plus false positives. The highest value is 1, and the lowest value is 0.

$$Precision = \frac{TP}{TP + FP} \tag{10}$$

4) F_1 SCORE

F_1 score conveys the balance between the precision and the recall; in other words, the f_1 score is the harmonic mean

between precision and recall. Like the other scores, f_1 has the same range of values from 1 to 0.

$$F_1 Score = 2 * \frac{Precsicion * Recall}{Precsicion + Recal} \tag{11}$$

This study used the parameters mentioned above to evaluate the performance of all algorithms. We compare algorithm accuracies and propose the best performer on the Shopify data.

V. RESULTS AND DISCUSSION

This section will discuss the result of the experiments for solving the classification problem. This study used different techniques, especially for feature selection. The Chi2 technique was used, which gave an improvement in our experiment results. We compare the results of two tree-based ensemble algorithms, RF, and AC, with a statistical algorithm, LR. As mentioned above, four popular evaluation scores were used to compare machine learning algorithms (see section IV(C)). This study set up different experiments to evaluate the leaning models by using features extraction techniques on the Shopify app dataset. In the first approach, we applied both feature extraction techniques (BoW, TF-IDF) with all machine learning algorithms to classify happy and unhappy users. The results are shown in Tables 11 and 12, where bold font illustrates the highest scores.

TABLE 11. Machine learning models results with BoW.

Models	Features	Accuracy	Precision	Recall	F_1 Score
LR	BoW	**0.80**	**0.80**	0.86	0.82
RF	BoW	0.79	0.79	**0.90**	**0.83**
AC	BoW	0.78	0.78	0.88	**0.83**

TABLE 12. Machine learning models results with TF-IDF.

Models	Features	Accuracy	Precision	Recall	F_1 Score
LR	TF-IDF	**0.81**	**0.82**	0.86	**0.84**
RF	TF-IDF	0.79	0.79	**0.88**	0.83
AC	TF-IDF	0.78	0.79	0.84	0.82

The accuracy scores of the learning models show that LR performs very well on the test dataset. LR gave high results in terms of f_1 score with respect to both feature extraction techniques (BoW and TF-IDF). RF achieves the highest recall score with both feature extraction techniques, while the AC also achieves acceptable results. According to the confusion matrix, LR with TF-IDF achieved the best result and also achieved the highest TP and TN rates. LR gives 3098 (1946 happy & 1152 unhappy) correct predictions and 723 (306 happy & 417 unhappy) wrong predictions against 3821 (2252 happy & 1569 unhappy) test examples with TF-IDF, which is higher than the others, as shown in the confusion matrix in Figure 6. The average accuracy for happy class is 86% and unhappy class is upto 73%. By changing feature engineering techniques, such as (BoW + Chi2) and (TF-IDF + Chi2) learning models improved their results. BoW was used to extract the features from text, and Chi2 was used to select the important features from the extracted

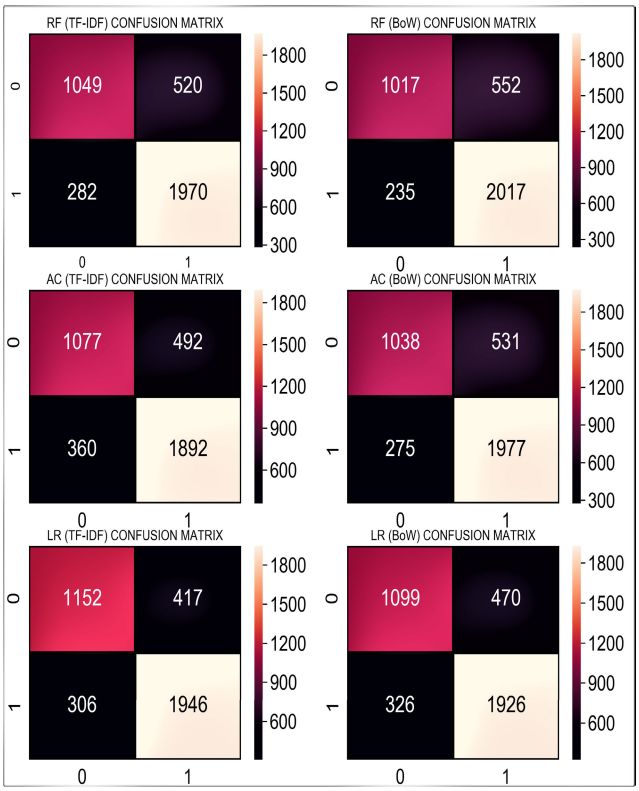


FIGURE 6. In this figure, confusion matrices of RF, LR, and AC are shown using the BoW and TF-IDF techniques. Here 0 represents the unhappy class whereas 1 represents the happy class.

TABLE 13. Machine learning models results with (BoW + Chi2).

Models	Features	Accuracy	Precision	Recall	F_1 Score
LR	BoW + Chi2	**0.80**	**0.82**	0.87	**0.84**
RF	BoW + Chi2	0.79	0.78	**0.89**	0.83
AC	BoW + Chi2	0.78	0.79	0.84	0.82

TABLE 14. Machine learning model results with (TF-IDF + Chi2).

Models	Features	Accuracy	Precision	Recall	F_1 Score
LR	TF-IDF + Chi2	**0.83**	**0.82**	0.88	**0.86**
RF	TF-IDF + Chi	0.80	0.79	**0.90**	0.84
AC	TF-IDF + Chi2	0.79	0.79	0.88	0.83

features. These techniques also gave us some better results with LR, the results in Table 13 and 14 show that LR also performs well under the (BoW + Chi2) and (TF-IDF + Chi2) feature engineering techniques. Again, bold font indicates the highest score values.

Results with (BoW + Chi2) are lower than (TF-IDF + Chi2) because the combination of TF-IDF and Chi2 gives more valuable features to learning models for fitting. LR gives more accurate predictions for happy and unhappy users with both feature engineering techniques. According to the confusion matrix (Figure 7), LR with the combination of TF-IDF and Chi2 gives 3186 (1992 happy & 1194 unhappy) correct predictions and 635 (260 happy & 375 unhappy) wrong predictions out of 3821 (2252 happy & 1569 unhappy) predictions with an average accruacy for happy class 88% and unhappy class upto 76%. These results clearly showing the increase of correct predictions and decrease of wrong

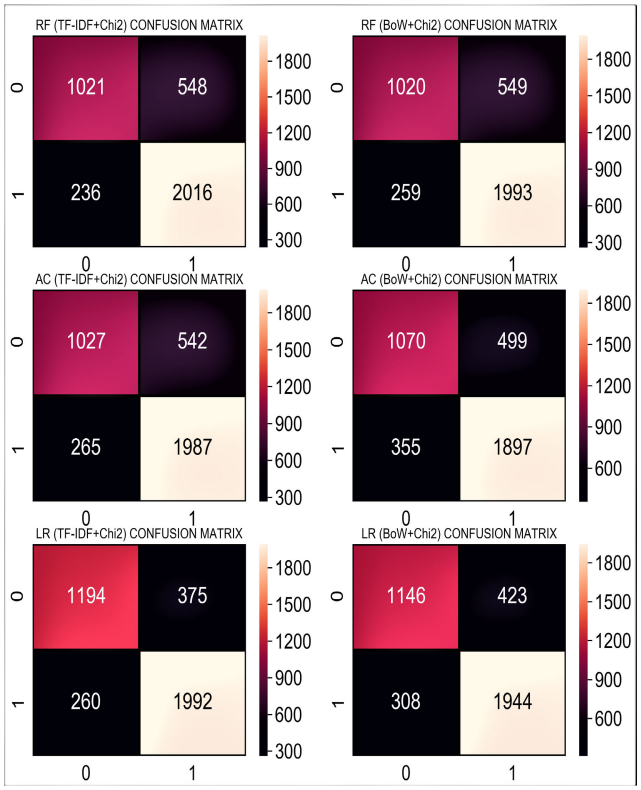


FIGURE 7. In this figure, confusion matrices of RF, LR, and AC are shown when using the (BoW+Chi2) and (TF-IDF+Chi2) techniques. Here 0 represents the unhappy class whereas 1 represents the happy class.

predictions by learning models when we use a combination of features. In this study, performance was measured by accuracy, precision, recall, and f_1 score for all approaches, but specifically, we use f_1 score to compare the performance of machine learning models because this score is mostly used for measuring the performance of supervised machine learning algorithms [44]. LR tops the table by achieving the highest scores, as illustrated in Figures 8, 9, 10, 11 and 12.

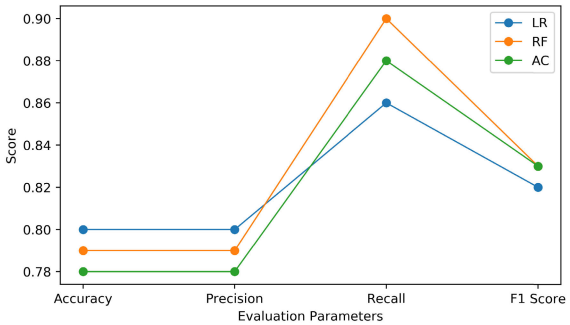


FIGURE 8. Comparison between accuracy, precision, recall, and f_1 score of all learning models using the BoW technique.

According to our comparisons, LR performed very well with all feature engineering techniques and beat the ensemble learning models during the classification of users. As mentioned above, this study had binary target classes (happy and

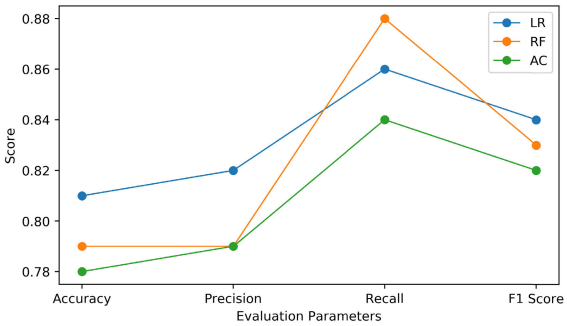


FIGURE 9. Comparison between accuracy, precision, recall, and f_1 score of all learning models using the TF-IDF technique.

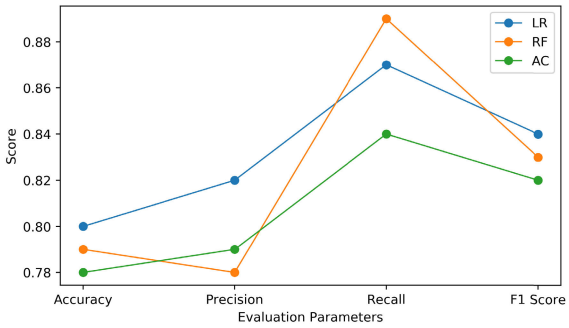


FIGURE 10. Comparison between accuracy, precision, recall, and f_1 score of all learning models using the (BoW + Chi2) technique.

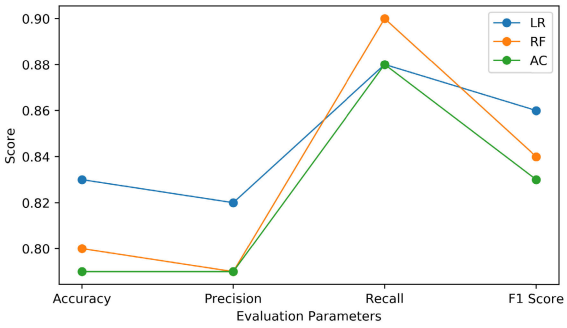


FIGURE 11. Comparison between accuracy, precision, recall, and f_1 score of all learning models using the (TF-IDF + Chi2) technique.

unhappy users), and according to the situation and problem, LR performs better than RF and AC because LR performs better when the prediction probability ranges from 0 to 1. In other words, LR performs well in binary classification problems with its "multi_class= ovr" parameter, as in our binary classification problem. AC also achieved acceptable results on text data because of its boosting method. The boosting method produces highly accurate predictions by combining many weak learners [45], but the boosting method leads toward overfitting the classifiers [31]. RF in our experiments achieved the highest recall score because it is ensemble learning in which using multiple decision trees often performs better than single decision trees in classification tasks [9], [46]. That is the reason we used an ensemble model in our experiments. Experimental results also show that features engineering techniques are effective in machine learning, and

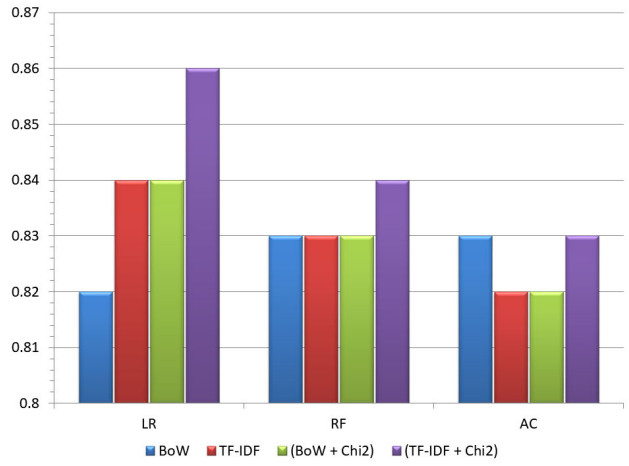


FIGURE 12. Comparison between f_1 scores of machine learning models using all feature engineering techniques.

these techniques are helpful in improving the accuracy of learning models. All learning models achieve their desired results because feature selection techniques give us important features from the extracted features, which is an effective way to increase the accuracy of learning models.

The performance of learning models without feature selection techniques (Chi2) and with simple feature extraction techniques, such as BoW and TF-IDF, is also acceptable. As a comparison between learning models, LR wins the race in all situations to solve user's classification problems. One can deploy the Wilcoxon test to further validate the performance of our porposed model is statistically significant.

VI. CONCLUSION

This study, for the very first time to the best of our knowledge, exploits the use of different machine learning approaches to solve user's review classification problems based on different feature engineering techniques, such as BoW, TF-IDF, and Chi2. The classifiers (RF, LR, and AC) were trained on text reviews to predict the user's review as being happy or unhappy for Shopify apps only. The comparative analysis reveals that LR outperformed in the case of using TF-IDF and Chi2 together. We end the conclusion by pointing out that the results and conclusion of our experiments are based on a single dataset (the Shopify app dataset), which was never used before for classification purposes, and these algorithms have not yet been tested on other datasets. It is possible that our results are specific to the dataset being used. Our future work entails testing deep machine learning models on different text and categorical datasets for the purpose of user review classification.

REFERENCES

[1] S. R. Das and M. Y. Chen, "Yahoo! For Amazon: Sentiment extraction from small talk on the Web," *Manage. Sci.*, vol. 53, no. 9, pp. 1375–1388, Sep. 2007.

[2] J. Chevalier and D. Mayzlin, "The effect of word of mouth on sales: Online book reviews," Nat. Bureau Economic Res., Cambridge, MA, USA, Tech. Rep. w10148, 2003.

[3] E. Guzman and W. Maalej, "How do users like this feature? A fine grained sentiment analysis of app reviews," in *Proc. IEEE 22nd Int. Requirements Eng. Conf. (RE)*, Aug. 2014, pp. 153–162.

[4] C. Iacob and R. Harrison, "Retrieving and analyzing mobile apps feature requests from online reviews," in *Proc. 10th Work. Conf. Mining Softw. Repositories (MSR)*, May 2013, pp. 41–44.

[5] W. Maalej and H. Nabil, "Bug report, feature request, or simply praise? On automatically classifying app reviews," in *Proc. IEEE 23rd Int. Requirements Eng. Conf. (RE)*, Aug. 2015, pp. 116–125.

[6] T. Pranckevičius and V. Marcinkevičius, "Comparison of naive Bayes, random forest, decision tree, support vector machines, and logistic regression classifiers for text reviews classification," *Baltic J. Mod. Comput.*, vol. 5, no. 2, pp. 221–232, 2017.

[7] E. Loper and S. Bird, "NLTK: The natural language toolkit," 2002, *arXiv:cs/0205028*. [Online]. Available: https://arxiv.org/abs/cs/0205028

[8] J.-H. Xue and D. M. Titterington, "Comment on 'on discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes,'" *Neural Process. Lett.*, vol. 28, no. 3, pp. 169–187, Dec. 2008.

[9] E. Guzman, M. El-Haliby, and B. Bruegge, "Ensemble methods for app review classification: An approach for software evolution (N)," in *Proc. 30th IEEE/ACM Int. Conf. Automat. Softw. Eng. (ASE)*, Nov. 2015, pp. 771–776.

[10] S. Panichella, A. Di Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall, "How can I improve my app? Classifying user reviews for software maintenance and evolution," in *Proc. IEEE Int. Conf. Softw. Maintenance Evol. (ICSME)*, Sep. 2015, pp. 281–290.

[11] S. M. Isa, R. Suwandi, and Y. P. Andrean, *Optimizing the Hyperparameter of Feature Extraction and Machine Learning Classification Algorithms*. London, U.K.: The Science and Information Organization, 2019.

[12] F. Rustam, I. Ashraf, A. Mehmood, S. Ullah, and G. Choi, "Tweets classification on the base of sentiments for US airline companies," *Entropy*, vol. 21, no. 11, p. 1078, Nov. 2019.

[13] V. Svetnik, A. Liaw, C. Tong, J. C. Culberson, R. P. Sheridan, and B. P. Feuston, "Random forest: A classification and regression tool for compound classification and QSAR modeling," *J. Chem. Inf. Comput. Sci.*, vol. 43, no. 6, pp. 1947–1958, Nov. 2003.

[14] F. F. Bocca and L. H. A. Rodrigues, "The effect of tuning, feature engineering, and feature selection in data mining applied to rainfed sugarcane yield modelling," *Comput. Electron. Agricult.*, vol. 128, pp. 67–76, Oct. 2016.

[15] J. Heaton, "An empirical analysis of feature engineering for predictive modeling," in *Proc. SoutheastCon*, Mar. 2016, pp. 1–6.

[16] S. C. Eshan and M. S. Hasan, "An application of machine learning to detect abusive Bengali text," in *Proc. 20th Int. Conf. Comput. Inf. Technol. (ICCIT)*, Dec. 2017, pp. 1–6.

[17] X. Hu, J. S. Downie, and A. F. Ehmann, "Lyric text mining in music mood classification," *Amer. Music*, vol. 183, no. 5, 049, pp. 2–209, 2009.

[18] B. Yu, "An evaluation of text classification methods for literary study," *Literary Linguistic Comput.*, vol. 23, no. 3, pp. 327–343, Sep. 2008.

[19] S. Robertson, "Understanding inverse document frequency: On theoretical arguments for IDF," *J. Document.*, vol. 60, no. 5, pp. 503–520, Oct. 2004.

[20] W. Zhang, T. Yoshida, and X. Tang, "A comparative study of TF*IDF, LSI and multi-words for text classification," *Expert Syst. Appl.*, vol. 38, no. 3, pp. 2758–2765, Mar. 2011.

[21] P. Meesad, P. Boonrawd, and V. Nuipian, "A chi-square-test for word importance differentiation in text classification," in *Proc. Int. Conf. Inf. Electron. Eng.*, 2011, pp. 110–114.

[22] S. Bird, "NLTK-Lite: Efficient scripting for natural language processing," in *Proc. 4th Int. Conf. Natural Lang. Process. (ICON)*, 2005, pp. 11–18.

[23] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," *Emerg. Artif. Intell. Appl. Comput. Eng.*, vol. 160, pp. 3–24, 2007.

[24] G. Biau and E. Scornet, "A random forest guided tour," *TEST*, vol. 25, no. 2, pp. 197–227, Jun. 2016.

[25] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, Aug. 1996.

[26] A. Liaw and M. Wiener, "Classification and regression by randomforest," *R News*, vol. 2, no. 3, pp. 18–22, 2002.

[27] J. Benediktsson and P. Swain, "Consensus theoretic classification methods," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, no. 4, pp. 688–704, 1992.

[28] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Trans. Pattern Anal. Mach. Intell.*, no. 10, pp. 993–1001, Oct. 1990.

[29] L. I. Kuncheva, "That elusive diversity in classifier ensembles," in *Proc. Iberian Conf. Pattern Recognit. Image Anal.* Berlin, Germany: Springer, 2003, pp. 1126–1138.

[30] R. E. Schapire, "A brief introduction to boosting," in *Proc. IJCAI*, vol. 99, 1999, pp. 1401–1406.
[31] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, Aug. 1997.
[32] D. S. Palmer, N. M. O'Boyle, R. C. Glen, and J. B. O. Mitchell, "Random forest models to predict aqueous solubility," *J. Chem. Inf. Model.*, vol. 47, no. 1, pp. 150–158, Jan. 2007.
[33] Y. Zhang, H. Zhang, J. Cai, and B. Yang, "A weighted voting classifier based on differential evolution," *Abstract Appl. Anal.*, vol. 2014, 2014.
[34] J. Belanich and L. E. Ortiz, "On the convergence properties of optimal Adaboost," 2012, *arXiv:1212.1108*. https://arxiv.org/abs/1212.1108
[35] I. Sevilla-Noarbe and P. Etayo-Sotos, "Effect of training characteristics on object classification: An application using boosted decision trees," *Astron. Comput.*, vol. 11, pp. 64–72, Jun. 2015.
[36] F. Elorrieta, S. Eyheramendy, A. Jordán, I. Dékány, M. Catelan, R. Angeloni, J. Alonso-García, R. Contreras-Ramos, F. Gran, and G. Hajdu, "A machine learned classifier for RR Lyrae in the VVV survey," *Astron. Astrophys.*, vol. 595, p. A82, Nov. 2016.
[37] R. Zitlau, B. Hoyle, K. Paech, J. Weller, M. M. Rau, and S. Seitz, "Stacking for machine learning redshifts applied to SDSS galaxies," *Monthly Notices Roy. Astronomical Soc.*, vol. 460, no. 3, pp. 3152–3162, 2016.
[38] A. Mayr, H. Binder, O. Gefeller, and M. Schmid, "The evolution of boosting algorithms," *Methods Inf. Med.*, vol. 53, no. 06, pp. 419–427, 2014.
[39] G. Grégoire, "Multiple linear regression," in *European Astronomical Society*, vol. 66. Les Ulis, France: EDP Sciences, 2014, pp. 45–72.
[40] F. Sebastiani, "Machine learning in automated text categorization," *ACM Comput. Surv.*, vol. 34, no. 1, pp. 1–47, 2002.
[41] B. Issac and W. J. Jap, "Implementing spam detection using Bayesian and porter stemmer keyword stripping approaches," in *Proc. IEEE Region Conf. (TENCON)*, Jan. 2009, pp. 1–5.
[42] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," 2016, *arXiv:1607.01759*. [Online]. Available: https://arxiv.org/abs/1607.01759
[43] Y. J. Huang, R. Powers, and G. T. Montelione, "Protein NMR recall, precision, and F-measure scores (RPF scores): Structure quality assessment measures based on information retrieval statistics," *J. Amer. Chem. Soc.*, vol. 127, no. 6, pp. 1665–1674, 2005.
[44] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," 2016, *arXiv:1606.05250*. [Online]. Available: https://arxiv.org/abs/1606.05250
[45] S. Mathanker, P. Weckler, T. Bowser, N. Wang, and N. Maness, "Adaboost classifiers for pecan defect classification," *Comput. Electron. Agricult.*, vol. 77, no. 1, pp. 60–68, 2011.
[46] A. C. Tan and D. Gilbert, "Ensemble machine learning on gene expression data for cancer classification," *Appl. Bioinf.*, vol. 2, no. 3, pp. S75–83, 2003.

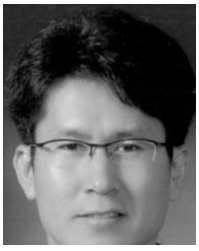
FURQAN RUSTAM received the M.C.S. degree from the Department of Computer Science, Islamia University of Bahawalpur, Pakistan, in October 2017. He is currently pursuing the master's degree in computer science with the Department of Computer Science, Khwaja Fareed University of Engineering and Information Technology (KFUEIT), Rahim Yar Khan, Pakistan. He is also serving as a Research Assistant with the Fareed Computing and Research Center, KFUEIT. His recent research interests are related to data mining, mainly working machine learning and deep learning-based IoT, and text mining tasks.

ARIF MEHMOOD received the Ph.D. degree from the Department of Information and Communication Engineering, Yeungnam University, South Korea, in November 2017. He is currently working as an Assistant Professor with the Department of Computer Science and IT, The Islamia University of Bahawalpur, Pakistan. His recent research interests are related to data mining, mainly working on AI and deep learning-based text mining, and data science management technologies.

MUHAMMAD AHMAD is currently an Assistant Professor with the Department of Computer Engineering, Khwaja Fareed University of Engineering and Information Technology, Pakistan. He is also associated with the Research Group, Advanced Image Processing Research Lab (AIPRL), the First Hyperspectral Imaging Lab, Pakistan. He is also associated with the University of Messina, Messina, Italy, as a Research Fellow. He has authored a number of research articles in reputed journals and conferences. He is a Regular Reviewer of Springer Nature journals, NCAA, the IEEE (TIE, TNNLS, TGRS, TIP, GRSL, GRSM, JSTARS, TMC, the TRANSACTIONS ON MULTIMEDIA, ACCESS, COMPUTERS, SENSORS, the TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING), *MDPI Journals*, *Optik*, *Measurement Science and Technology*, *IET Journals*, and the *Transactions on Internet and Information Systems*. His current research interests include machine learning, computer vision, remote sensing, hyperspectral imaging, and wearable computing.

SALEEM ULLAH was born in Ahmedpur East, Pakistan, in 1983. He received the B.Sc. degree in computer science from Islamia University Bahawalpur, Pakistan, in 2003, the M.I.T. degree in computer science from Bahauddin Zakariya University, Multan, in 2005, and the Ph.D. degree from Chongqing University, China, in 2012. From 2006 to 2009, he worked as a Network/IT Administrator in different companies. From August 2012 to February 2016, he worked as an Assistant Professor with Islamia University Bahawalpur. He has been working as an Associate Professor with the Khwaja Fareed University of Engineering and Information Technology, Rahim Yar Khan, since February 2016. He has almost 13 years of Industry experience in the field of IT. He is also an active Researcher in the field of adhoc networks, congestion control, and security.

DOST MUHAMMAD KHAN received the M.Sc. degree (Hons.) in computer science from Bahauddin Zakariya University (BZU), Multan, in 1990, and the Ph.D. degree from the School of Innovative Technologies and Engineering (SITE), University of Technology, Mauritius (UTM), in 2013. He is currently working as an Assistant Professor with the Department of Computer Science and IT, The Islamia University of Bahawalpur, Pakistan. His areas of research are data mining and data mining techniques, multiagent system (MAS), object-oriented data base systems, and formal methods in software engineering.



GYU SANG CHOI received the Ph.D. degree from the Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA, USA, in 2005. He was a Research Staff Member with the Samsung Advanced Institute of Technology (SAIT) for Samsung Electronics, from 2006 to 2009. Since 2009, he has been a Faculty Member with the Department of Information and Communication, Yeungnam University, South Korea. His research areas include non-volatile memory and storage systems.