Journal of Information Technology and Computer Science
Volume 6, Number 1, April 2021, pp. 33-49
Journal Homepage: www.jitecs.ub.ac.id

Comparison of Bagging Ensemble Combination Rules for Imbalanced Text Sentiment Analysis

Reiza Adi Cahya*[1], Fitra A. Bachtiar[2], Wayan Firdaus Mahmudy[3]

[1,2,3]Faculty of Computer Science, University of Brawijaya
[1]reizaadi@student.ub.ac.id, [2]fitra.bachtiar@ub.ac.id, [3]wayanfm@ub.ac.id
*Corresponding Author

Received 08 June 2020; accepted 09 April 2021

Abstract. The wealth of opinions expressed by users on micro-blogging sites can be beneficial for product manufacturers of service providers, as they can gain insights about certain aspects of their products or services. The most common approach for analyzing text opinion is using machine learning. However. opinion data are often imbalanced, e.g. the number of positive sentiments heavily outnumbered the negative sentiments. Ensemble technique, which combines multiple classification algorithms to make decisions, can be used to tackle imbalanced data to learn from multiple balanced datasets. The decision of ensemble is obtained by combining the decisions of individual classifiers using a certain rule. Therefore, rule selection is an important factor in ensemble design. This research aims to investigate the best decision combination rule for imbalanced text data. Multinomial Naïve Bayes, Complement Naïve Bayes, Support Vector Machine, and Softmax Regression are used for base classifiers, and max, min, product, sum, vote, and meta-classifier rules are considered for decision combination. The experiment is done on several Twitter datasets. From the experimental results, it is found that the Softmax Regression ensemble with meta-classifier combination rule performs the best in all except in one dataset. However, it is also found that the training of the Softmax Regression ensemble requires intensive computational resources.

Keyword :ensemble, SUM, SR, classifier, dataset

1. Introduction

It is common to see relatives or friends express their stance on the social phenomenon or certain products or services in social media timelines. Not only on social media but also personal blogs have also become a way for people to articulate their ideas. Other than in personal platforms like social media and blogs, commercial websites also provide a place for their users to write reviews about products or services they use. Internet forums have also accommodated like-minded people to discuss their topic-of-interest. Indeed, ease of communication provided by Internet technologies has allowed people to freely express their opinion regarding a certain topic [1].

The abundant amount of opinions provided on these platforms has many advantages for various parties. For customers, opinions can provide additional information in assisting financial decisions. For example, users who want to buy a certain product may refer to existing user reviews to see if the product fulfills their needs. For manufacturers of service providers, these opinions can be used to gather public reception regarding their

products/services. Analyzing the opinions can reveal positive and negative aspect of the product, and decision makers can devise a correct measure to improve their product [2].

Analysis of opinions falls into a field of research called sentiment analysis (SA). SA deals with the recognition of one's emotion toward a certain topic, products, or phenomenon [2]. The emotion can be one's polarity, such as if one is positive toward the topic, negative, or neutral. Emotion can also be identified as expressions such as happiness, sadness, confession, etc. The analysis of sentiments can also be performed at various levels of granularity. The broadest, and the least detailed, analysis is done at the document level. Here, the whole document, that can be in the form of a blog post, news articles, or single Tweet, is classified into single sentiment. Finer levels of granularity can be performed at paragraph- and sentence-level analyses. The most detailed analysis is done at the aspect level, where each aspect word is classified into sentiment.

To perform SA at various, there are multiple approaches such as lexicon-based or machine learning-based approaches. However, the most used approach today is the use of machine learning [3]. Machine learning approaches have strived because of the abundance of the openly-available labeled dataset [1]. Machine learning approaches use various algorithms such as Support Vector Machines or Neural Networks to learn from the dataset and build a classification model to analyze the sentiment of an unknown document [4][5][6].

As with any approach, machine learning has to deal with challenges. The size of the dataset is often very big. Also, it is common for the dataset to have thousands to ten-thousands of documents. Moreover, data must also be represented in some form of features to be used in machine learning algorithms. Larger documents lead to a larger number of words that need to be represented in some form of numerical data [7]. Therefore, machine learning algorithms must be able to learn from very high dimensional data.

Another important factor is that real-life data is often imbalanced [8]. In classification problems such as SA, an imbalance is a condition where a class label heavily outnumbers other class labels. For example, in product reviews, the number of positive reviews may heavily outnumber negative or neutral reviews. The presence of majority class label and minority class labels may provide difficulties for machine learning algorithms, as minority class label, which is in a smaller number of documents, may be treated as noise [4].

Multiple approaches have been proposed to handle imbalanced data. Re-sampling techniques attempt to balance the number of samples in the dataset. Balancing can be done by increasing the number of minority class samples (over-sampling) or decreasing the number of majority class samples (under-sampling) [9]. The most used methods include synthetic minority over-sampling technique (SMOTE) [10] and random under-sampling (RUS) [11]. Resampling methods are straightforward to implement [4]. However, over-sampling is prone to overfitting, and under-sampling has the probability to discard important data points [12].

Another common approach is to use ensemble classifiers. Ensemble methods combine several basic classifiers to build a model with better recognition capability [4]. With the analogy of consulting several experts to get a more informed decision, a combination of several classifiers helps reduce the variance and bias of a single classifier. Multiple classifiers can also learn more complex decision boundaries that may be too difficult for a single classifier to understand. The ensemble can also break down a large imbalanced dataset into partitions of balanced subsets, building a model for each data subset [13][14].

Based on the problem of imbalanced data in real-world text datasets and ensemble methods to tackle the problem, this research aims to evaluate the effectiveness of various ensemble rules. The runtime of the classifier training phase is also examined, as multiple classifiers must be trained to form the ensemble model. In this paper, Naïve Bayes (NB) is used as classifiers because some models of NB are specifically created for text classification and have shown good results in previous studies [15][16][17][18]. Support Vector Machines (SVM) and Sofmax Regression (SR) models are also used as both are considered suitable for text processing [1][19][20].The paper is organized as follows: several ensemble methods

are described in Section 2. The methodology used in this research is explained in Section 3. The result is discussed in Section 4 and conclusions can be found in Section 5.

2. Related Works

2.1 Literature Review

In this section, several studies in the fields of ensembles, text processing, sentiment analysis, and imbalanced problems are reviewed.

Ensemble method is used alongside evolutionary search Particle Swarm Optimization (PSO) to perform sentiment analysis for two imbalanced multiclass datasets [1]. PSO is used to select text features, which includes words, part-of-speech, named entity, etc. Each particle in the swarm represents the selected feature space, which then used to train a classifier. Several top-*n* classifiers are used to construct the ensemble. PSO is used again to selectively drop classifiers from the ensemble. The method shows improvement over existing methods.

Another text processing method based on feature weighting and Naïve Bayes (NB) is proposed [21]. Feature weighting is used to alleviate the weakness of feature independence assumption of NB. The method uses correlation-based feature selection (CFS) to select features, and selected features have increased weight. The weight is used to modify the decision function and probability estimation of NB. Modified NB models Multinomial NB (MNB), Complement NB (CNB), and One-versus-all NB (combination of MNB and CNB) is used with the modified weights. The experiments are conducted on both text and non-text databases. Modified weights have been shown to improve classification results.

Evolutionary algorithm is also used to modify NB method for text classification [16]. The method converts probability estimation problem of NB into optimization problem. Differential evolution (DE) is used to find the optimal estimated conditional probability. DE is also enhanced with multi-parent and crossover methods. The proposed method is shown to perform better than classical classifiers on multiple text datasets.

Another evolutionary algorithm based on cuckoo search (CS) is also used for Twitter sentiment analysis [22]. Twitter datasets are pre-processed into features such as total number of words, negative emoji, and positive emoji. CS is used to optimize cluster centroid based on maximization of inter-class variance. After the best cluster centroid is found, the classification is carried out using *k*-means algorithm. Experiments on four Twitter datasets shows improvement compared to evolutionary algorithms such as PSO and DE.

Deep learning models is also suitable for text classification [20]. Deep learning models constructs new feature representation from sparse high-dimensional text representations such as TFIDF. The study uses deep belief network (DBN) to learn the new feature space. Softmax Regression (SR) is used to classify the learned feature representation from DBN. Initially, the DBN and SR are trained independently. Then, both models are combined and trained together to get better model. The experiment compares gradient descent and L-BFGS methods to train the network. It is found that L-BFGS perform better than gradient descent.

Ensemble method is used to classify imbalanced dataset [9]. The method chooses appropriate ensemble, feature selection, classifier, and ensemble rule for different datasets. Included ensemble methods are oversampling-based bagging, undersampling-based bagging, and AdaBoost M1. Included feature selection methods are wrapper-based PSO and filter-based fast-correlation based feature selection (FCBF). Used classifiers are C4.5, SVM, radial basis neural network (RBF-NN), data gravitation classifier (DGC), and *k*-nearest neighbor (KNN). The combination of methods is then selected based on the number of samples, number of classes, number of features, and the imbalance rate. The proposed method shows better performance than existing ensemble methods on a small database.

An ensemble method with modified rules is also introduced [14]. The ensemble method splits the dataset using two schemes i.e., splitbal and clusterbal. Splitbal creates multiple balanced datasets by splitting data randomly, while clusterbal uses clustering. The modified ensemble rules incorporate the distance of test sample to the training samples, assigning

higher weight to classifier with nearer samples. Experiments on non-text datasets shows that splitbal performs better more often than clusterbal and other ensemble methods, and modified max rule is often the best combination rule for the ensembles.

2.2 Bagging Ensemble

Breiman's Bagging (bootstrap aggregating) [23] is the simplest ensemble model and very easy to implement. Each model in the bagging ensemble is trained using a fraction of data and/or feature subset from the dataset. As every model is independent of each other, it can also be trained in parallel [4]. The pseudocode of bagging can be viewed in Procedure 1.

The classification of a new test sample of the ensemble is determined by combining the decision of classifiers (i.e., the scores of new samples belonging to certain class). The possible combination rules are max rule, min rule, product rule, sum rule, and majority vote [14] or with meta-classifier [24]. The formulations of combination rules are listed in Table 1, with S_{ic} refers to normalized score for class c in classifier f_i and R_c refers to the ensemble score of class c. The $1(c = v_i)$ operator has value of 1 if the specified condition is fulfilled, and 0 if not. The final decision is the class with the highest R_c.

2.3 Multinomial Naïve Bayes

Multinomial Naïve Bayes (MNB) is one of the state-of-the-art Naïve Bayesian models for text classification, with the other being complement naïve Bayes (CNB) [17]. MNB is an NB model that allows for term frequency vector representation. MNB classifies test document $\boldsymbol{x} = [x_1, x_2, \dots, x_m]$ using decision function written in Equation (1),

$$f(\boldsymbol{x}) = \underset{1 \le c \le k}{\operatorname{argmax}} [\textstyle\sum_{i=0}^{m} x_i \log P(t_i|c)] \tag{1}$$

where k is the number of classes in the dataset, x_i is the term frequency for i-th word in document $\boldsymbol{x}$ m is the size of the dictionary, and $\log P(t_i|c)$ is the conditional probability of term i given class c. The conditional probability is computed in Equation (2)

$$P(t_i|c) = \frac{1+\sum_{j=1}^{m} x_{ji} 1(c_j=c)}{m+\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ji}\delta(c_j=c)} \tag{2}$$

where n is the total number of documents. Eq. (2) calculates the sum of the frequency of term i in all documents belonging to class c, divided by the sum of frequencies of all terms in all documents belonging to class c (smoothing is applied to avoid division by zero).

Procedure 1 Bagging
Input: Sequence of samples S, learning algorithm $WeakLearn$, number of classifiers N_{clf}
1. **For** $t = 1,2, \dots, N_{clf}$ **do**
2. S_t = select random subset of training data and/or feature space from S.
3. f_t = train $WeakLearn$ using S_t
4. Add f_t to ensemble
5. **End for**

Output: Ensemble of classifiers $f_1, f_2, \dots, f_{N_{clf}}$

Fig. 1 Pseudocode for Bagging

Table 1. Combination Rules for Bagging Ensemble

Rule	Score for Each Class	Description
Max	$R_c = \underset{0 \le i < N_{clf}}{\arg\max} S_{ic}$	Score of each class is the highest normalized score from all classifiers
Min	$R_c = \underset{0 \le i < N_{clf}}{\arg\min} S_{ic}$	Score of each class is the lowest normalized score from all classifiers
Product	$R_c = \prod_{i=0}^{N_{clf}} S_{ic}$	Score of each class is the product of normalized scores of all classifiers
Sum	$R_c = \sum_{i=0}^{N_{clf}} S_{ic}$	Score of each class is the sum of normalized scores of all classifiers

Rule	Score for Each Class	Description
Vote	$R_c = \sum_{i=0}^{N_{clf}} 1(c = v_i)$, v_i = class with the highest score in f_i	Each class gets one vote if it is the class with the highest normalized score in a classifier
Clf	-	Train a meta-classifier using the scores of each classifier in the ensemble.

2.4 Complement Naïve Bayes

CNB is another NB state-of-the-art model that can be used for text classification [25]. CNB uses term frequencies to calculate probabilities like MNB. CNB, however, calculates conditional probabilities using the complement of class c ($\bar{c}$), i.e. all classes other than c.

The decision of CNB is written in Equation (3). It should be noted that the decision function of CNB differs from MNB, as CNB chooses the class that has the smallest score (using the minus sign).

$$f(\boldsymbol{x}) = \underset{1 \le c \le k}{\operatorname{argmax}} \left[-\sum_{i=0}^{m} x_i \log P(t_i|\bar{c}) \right] \tag{3}$$

$P(t_i|\bar{c})$ is the conditional probability of term i given all classes other than c. It is calculated using Equation (4).

$$P(t_i|\bar{c}) = \frac{1 + \sum_{j=1}^{m} x_{ji} 1(c_j \neq c)}{m + \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ji} \delta(c_j \neq c)} \tag{4}$$

2.5 Linear Support Vector Machine

Stochastic Gradient Descent (SGD)-based methods are suitable classification models for large-scale learning such as text classification [26]. SGD classifier has a decision function $f_{\boldsymbol{W}}(\boldsymbol{x})$ that is parameterized by weights $\boldsymbol{W}$. The weights $\boldsymbol{W}$ are chosen so that they minimize a loss function $L(\boldsymbol{W}; \boldsymbol{x}_i, y_i)$ over given dataset $S = [(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_n, y_n)]$. The value of $\boldsymbol{W}$ is gradually updated based on the gradient of the average loss function on dataset $\boldsymbol{S}$. Gradient descent becomes stochastic when only a small fraction of the dataset is used to estimate the gradient . The model of linear SGD classifier can be viewed in Fig. 2. Inputs $x_1, x_2, x_3, \dots, x_n$ is the features of the test data $\boldsymbol{x}$. Weights $\boldsymbol{W}$ is used to with the input to btain classifier outputs s_1 through s_k, where k represents the number of classes. Linear Support Vector Machine (SVM) can be formulated as an SGD classifier using the multi-class hinge loss as the loss function. Additionally, SVM also uses L2 regularization into the loss function to prevent overfitting of the model.

Performing the training of SVM model requires updating the weight values. Initially the weight values are set randomly. Then, a subset of the dataset (a batch) is selected randomly an is used to calculate the value of the loss function. Value of the loss function is used to get the gradient of currently trained batch. Then the weights are updated based on the gradient and the learning rate. The weight update process is repeated after a certain number of iterations. After the training is complete, test data sample can be classified based on the class with the highest score ($s_1, s_2, \dots, s_k$ in Fig. 2).

2.6 Softmax Regression

Another model for large-scale text classification is the Softmax Regression (SR) model [20], [27]. Softmax regression is the multi-class generalization of the Logistic Regression model. Like SVM, it is parameterized by weight matrix $\boldsymbol{W}$, and can be optimized using SGD procedure to minimize its loss function. Unlike SVM that use hinge loss, SR uses the cross-entropy (CE) loss with the L2 regularization. CE loss causes the output of the classifier to be the probability estimate of a data sample $\boldsymbol{x}$ belonging to certain class c. In other words, the value of s_c in Fig. 2 is $s_c = p(y = c \mid \boldsymbol{x})$. Based on the properties of probability, $s_1, s_2, \dots, s_k$ all are in the range of [0,1] and the sum of all scores equals to 1.

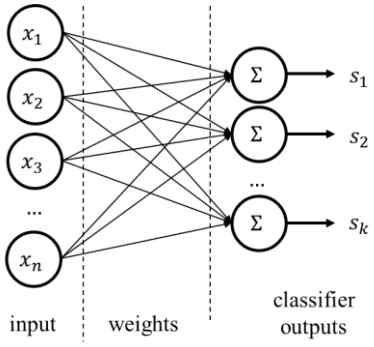

Fig. 2 Linear Classifier Model

3. Methodology

In this section the methodology of the research is explained. The overall process of the classification process is illustrated in Fig. 3. Initially, the imbalanced dataset is partitioned into multiple balanced subsets. Then, each balanced subset is used to train a classifier to create the models inside the ensemble. New test sample is classified using all models in the ensemble, each producing classifier results. All results are combined using a certain combination rule to obtain final classifier decision. The rest of this section explain the dataset, classification process, performance evaluation and implementation details.

3.1 Dataset Description

To perform the comparison of combination rules, experiments are carried out on several publicly-available datasets. Each dataset differs in the number of classes/sentiments and the imbalance ratio (IR). IR is the ratio between majority and minority class. The data distribution and the IR of each dataset can be seen in Table 2.

The brief explanation of each dataset is as follows:

1. Twitter-sanders-apple2[1]: binary classification (positive and negative) Twitter dataset.
2. Twitter-sanders-apple3[2]: the first dataset augmented with a neutral class.

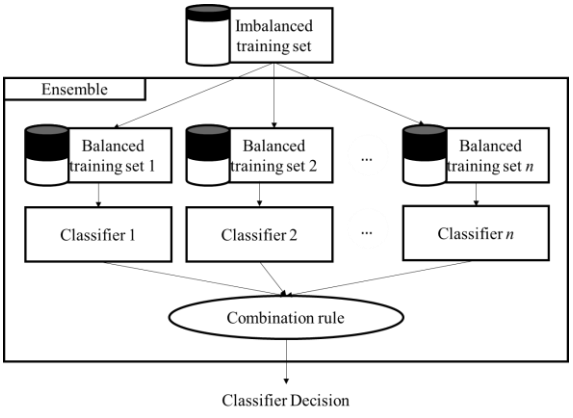

Fig. 3 Imbalanced Dataset Classification Using Ensemble

[1] http://boston.lti.cs.cmu.edu/classes/95-865-K/HW/HW3/
[2] http://boston.lti.cs.cmu.edu/classes/95-865-K/HW/HW3/

Table 2 Distribution of the Class Labels

Dataset	Very negative	Negative	Neutral	Positive	Very positive	Total	IR
Apple2	-	313	-	159	-	472	1.97
Apple3	-	313	499	159	-	971	3.14
Testdata	-	176	138	180	-	494	1.30
Airline	-	9159	3052	2353	-	14564	3.89
Self-driving	110	685	4245	1444	459	6943	38.59

3. Testdata-manual[3]: three-class dataset with fairly balanced classes.
4. Airline[4]: three-class dataset with a large amount of data.
5. Self-driving[5]: extremely imbalanced five-class dataset. There are tweets labeled with 'not relevant' in the original dataset. These tweets are not used in the experiment.

3.2 Text Preprocessing

For the tweets to be used in classification with the ensemble method, they need to be cleaned using a series of preprocessing steps. The preprocessing steps aim to standardize the tweet text. After the tweets are preprocessed, the numerical features can be extracted and the dataset is ready to be used in classification.

In this research, the performed preprocessing steps are:

1. The tweets are transformed into lower-case.
2. Contractions are expanded e.g. 'they're' to 'they are'.
3. Twitter usernames beginning with '@' are removed.
4. URL (starting with `https://`) in tweets are replaced with string '`url`' to prevent links from being converted into terms.
5. Hashtags are removed.
6. Non-alphabetic characters and additional whitespaces are removed from the tweets.
7. Words in tweets are lemmatized into the base form e.g. 'driving' to 'drive'.
8. Stop-words are removed.

After preprocessing, the numerical features can be extracted using term frequency TF weighting. The extracted TF features are then normalized such that the maximal absolute value of each word/feature will be 1.0. This normalization scheme ensures the sparsity of the data as a sparse data format is crucial to minimize the running time of the classifiers.

3.3 Classification with Bagging Ensemble

The dataset with TF-IDF features must be split into training and testing data using a 4-fold cross-validation method. The training is used to create the ensemble model, and consists of 75% instances of the dataset. The testing data is used to validate the performance of the trained ensemble model and consists of 25% instances of the dataset.

The training data are then further partitioned to create multiple classifier models that make up the ensemble. SplitBal [14] method is used to create multiple class-balanced partitions. The original SplitBal method was intended for a binary classification problem. To extend the method to a multi-class classification problem, first the minority class is determined to be the class with the least number of instances, e.g. the very negative class. Then, other classes are split into bins that have the same number of instances as the minority class. The number of created partitions equals the ratio between the number of instances in the majority class and minority class. In other words, the number of classifiers in the ensemble equals to the imbalance rate (IR) of the dataset. IR is the ratio between the amount of data in the class with most samples and amount of data in the class with least samples.

[3] http://help.sentiment140.com/for-students/
[4] https://www.kaggle.com/crowdflower/twitter-airline-sentiment
[5] https://www.kaggle.com/c/twitter-sentiment-analysis-self-driving-cars

Hence, the according to Table 2, ensemble for apple2 uses 2 classifiers, ensemble for apple3 uses 4 classifiers, testdata uses 2 classifiers, and self-driving uses 39 classifiers (IR numbers are rounded up). The ensemble is constructed using the same classifier. The meta-classifier also uses the same classifier as the ensemble members.

The created partitions are used to train individual classifiers. In this research, the ensemble is trained with classifiers of the same type, and there are four types of classifiers to be considered. The four classifiers are MNB, CNB, SVM, and SR. The MNB and CNB models do not have user-defined hyper-parameters. The SVM and SR have several parameters, i.e., learning rate/step size η, number of iterations $Niter$, regularization parameter λ. Grid search is used to find the best parameter for each rule. For grid search, SVM and SR use a batch size of 256. Additionally, data scaling is also treated as a parameter. The values of the parameters in the grid search are presented in Table 3.

Additionally, the classifier scores for MNB, CNB, and SVM needs to be adjusted. The combination rules presented in Table 1 requires classifiers to produce scores for each class given test data $\boldsymbol{x}$. The scores need to be in the range of $[0, 1]$ and all scores must sum to 1. However, only SR inherently supports producing normalized scores for each class, while other classifiers only produce raw scores for each class. Therefore, an additional step is taken to normalize the outputs of MNB, CNB, and SVM classifiers. The softmax function is a common method to normalize the raw scores [13]. The raw and normalized scores for NB classifiers are described in Table 4. The normalization process of the SVM classifier is illustrated in Fig. 4. It should be noted that normalized scores for NB and SVM are not probability estimates, as NB and SVM are known to be uncalibrated classifiers [28].

3.4 Performance Evaluation

To evaluate the classification model, F1 measure is used. F1 can be used in a multi-class classification problem using the macro-averaging [29]. The F1 score for a binary problem is formulated in (5). The macro-averaging of F1 is shown in (6). Macro-averaging is calculated by treating a class as a positive sample and other classes as negative. The process is repeated for each class as the positive class, and the F1-macro score is averaged from all individual F1scores.

$$F1 = \frac{2 \times precision \times recall}{precision + recall} \tag{5}$$

$$F1_{macro} = \frac{1}{K} \sum_{0 \le i \le K} F1_i \tag{6}$$

The stratified 4-fold cross-validation method is used in this research. Four partitions with similar class distribution are created. Three partitions are used as training to create the ensemble model. The remaining partition is used as testing data to validate the performance of the ensemble using the F1-macro score. The process is repeated so that all partition is used as testing data and F1-macro scores for four folds are obtained. The overall performance of the ensemble is the average F1-macro from 4-folds.

Table 3. Parameters for Grid Search

Categories	Parameters	Values
Preprocessing	Data scaling	TRUE, FALSE
MNB/CNB	-	-
SVM/SR	Batch size	256
	Learning rate η	0.001, 0.01, 0.1, 1
	Number of iterations $Niter$	500, 1000, 2000
	Regularization parameter λ	1e-7, 1e-6, 1e-5, 1e-4

Table 4. Normalization of Classifier Scores for NB Classifiers

Classifier	Raw score for class c	Normalized Score
MNB	$s_c = \sum_{i=0}^{m} x_i \log P(t_i \mid c)$	$s'_c = \frac{\exp(s_c)}{\sum_{i=1}^{k} \exp(s_i)}$
CNB	$s_c = -\sum_{i=0}^{m} x_i \log P(t_i \mid \bar{c})$	$s'_c = \frac{\exp(s_c)}{\sum_{i=1}^{k} \exp(s_i)}$

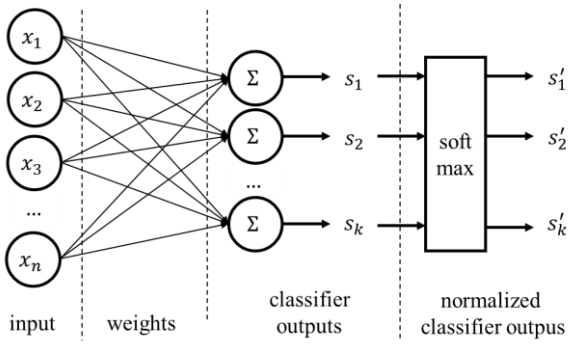

Fig. 4 Normalized Classifier Output for SVM

3.5 Implementation Details

The experiments are executed in Python 3.6.5 environment on top of Windows 10 system with Inter® Core™ i5-7200U CPU 2.5 GHz processor. The packages `numpy` and `scipy.sparse` are used for dense and sparse matrix computations, respectively. `nltk` is used for lemmatization, and `sklearn` is used for 4-fold CV and evaluation of the models.

4. Results and Discussion

4.1 Classification Results

The results of the experiments are presented in this section. The summary of the classification results is presented in Table 5. For each dataset, the results of the best-performing parameters for each classifier-rule combination are presented. Additionally, the base, non-ensembled classifier is also presented (model with '-' value in rule column) to decide whether the ensemble methods improve the original classifier.

The results from Twitter-sanders-apple2 dataset are presented in Table 6. It is divided into groups for each classifier. The italicized entry represents the base classifier, while the bold-faced entry represents the best performing classifier in its group.

The results show that for Naïve Bayesian methods, the ensemble models can improve the recognition ability of the base classifier, regardless of the rule. The best performing rule is the 'clf' (meta-classifier) rule for MNB. MNB models are known to be unsuitable for imbalanced class problems [25]. The ensemble model allows MNB to alleviate this problem by dividing the dataset into partitions of equal distribution between classes [14]. For CNB, all ensemble rules also improve the performance of the base classifier.

Table 5 Summary of Classification Results

Database	Best rule	Best classifier
twitter-sanders-apple2	Meta-classifier	Softmax Regression
twitter-sanders-apple3	Meta-classifier	Softmax Regression
testdata-manual	Meta-classifier	Softmax Regression
airline	Meta-classifier	Softmax Regression
self-driving	Non-ensemble (single classifier)	Softmax Regression

Table 6. Classification Results for Twitter-sanders-apple2 Dataset

classifier	scaling	Parameter	rule	f1-macro
MNB	*FALSE*	-	-	*74.21343*
MNB	TRUE	-	max	76.81697
MNB	TRUE	-	min	76.81697
MNB	TRUE	-	product	76.81697
MNB	TRUE	-	sum	76.81697
MNB	FALSE	-	vote	77.46156
MNB	**FALSE**	**-**	**clf**	**78.06839**
CNB	*TRUE*	-	-	*74.75359*
CNB	TRUE	-	max	76.81697
CNB	TRUE	-	min	76.81697
CNB	TRUE	-	product	76.81697
CNB	TRUE	-	sum	76.81697
CNB	**FALSE**	**-**	**vote**	**77.3864**
CNB	TRUE	-	clf	76.52934
SVM	*FALSE*	*η: 0.1, Niter: 500, λ: 1e-06*	-	*76.46366*
SVM	TRUE	η: 0.01, Niter: 500, λ: 0.0001	max	74.51911
SVM	TRUE	η: 0.01, Niter: 500, λ: 0.0001	min	74.51911
SVM	TRUE	η: 0.01, Niter: 500, λ: 0.0001	product	74.51911
SVM	TRUE	η: 0.01, Niter: 500, λ: 0.0001	sum	74.51911
SVM	TRUE	η: 0.01, Niter: 2000, λ: 1e-06	vote	78.49521
SVM	**FALSE**	**η: 0.1, Niter: 1000, λ: 1e-07**	**clf**	**79.45939**
SR	*FALSE*	*η: 0.1, Niter: 2000, λ: 1e-05*	-	*77.48613*
SR	TRUE	η: 0.1, Niter: 1000, λ: 1e-07	max	76.17287
SR	TRUE	η: 0.1, Niter: 1000, λ: 1e-07	min	76.17287
SR	TRUE	η: 0.1, Niter: 1000, λ: 0.0001	product	76.17287
SR	TRUE	η: 0.1, Niter: 1000, λ: 1e-06	sum	76.17287
SR	TRUE	η: 0.1, Niter: 1000, λ: 1e-06	vote	79.0912
SR	**FALSE**	**η: 1, Niter: 500, λ: 0.0001**	**clf**	**80.53566**

For the SGD-based algorithms (SVM/SR), only 'vote' and 'clf' rules can improve the performance, while other rules degrade the performance. Among the two rules, 'clf' is the better rule for both SVM and SR. The 'clf' rule uses a meta-classifier trained using the outputs of ensemble members [24]. Therefore, 'clf' rule can better map the outputs of the classifiers, and learn which ensemble members that can correctly classify a new data [13].

Among all the models tested for this dataset, the best performing model in this dataset is the SR ensemble with the 'clf' rule. The SR model has the learning rate $\eta = 1$. This means that each step of the SGD is noisier. The ensemble method can improve the performance of an unstable method. The decision boundary of an unstable model can vary greatly, which ensemble can exploit to select a better boundary for the overall system [13].

The results from Twittter-sanders-apple3 are presented in Table 7. This dataset is similar to the previous dataset, with a neutral class added. The addition of class turned binary classification into a multi-class classification which can significantly improve classification difficulty [4]. The performance degradation of all models can be seen, based on the comparison from Table 6 and Table 7. Similar to the previous dataset, all four classifiers also still gain performance improvements. Ensemble with the 'clf' rule is also the best overall performer. It also still holds that SGD models outperform the NB model, with SR being the best overall classifier. MNB and CNB, however, still gain better performance from ensemble formation compared to SGD models. MNB gained 3% performance while MNB gained 2% performance.

Table 7. Classification Results for Twitter-sanders-apple3 Dataset

classifier	scaling	Parameter	Rule	f1-macro
MNB	*FALSE*	-	-	*56.6087*
MNB	FALSE	-	Max	54.6941
MNB	FALSE	-	Min	52.648
MNB	FALSE	-	product	53.2478
MNB	FALSE	-	Sum	53.8036
MNB	FALSE	-	Vote	53.9041
MNB	**FALSE**	**-**	**Clf**	**59.7417**
CNB	*FALSE*	-	-	*57.3778*
CNB	FALSE	-	Max	56.3017
CNB	FALSE	-	Min	55.7564
CNB	FALSE	-	product	56.5984
CNB	FALSE	-	Sum	56.2305
CNB	FALSE	-	vote	55.018
CNB	**FALSE**	**-**	**clf**	**59.6351**
SVM	*FALSE*	*η: 0.1, Niter: 500, λ: 1e-05*	-	*59.6766*
SVM	TRUE	η: 0.01, Niter: 2000, λ: 0.0001	max	56.6346
SVM	TRUE	η: 0.01, Niter: 1000, λ: 1e-07	min	54.0625
SVM	FALSE	η: 0.01, Niter: 2000, λ: 1e-06	product	56.5514
SVM	FALSE	η: 0.01, Niter: 2000, λ: 0.0001	sum	56.624
SVM	TRUE	η: 0.01, Niter: 2000, λ: 1e-06	vote	55.6816
SVM	**FALSE**	**η: 0.1, Niter: 1000, λ: 1e-06**	**clf**	**60.8156**
SR	*TRUE*	*η: 1, Niter: 500, λ: 0.0001*	-	*60.5888*
SR	TRUE	η: 0.1, Niter: 1000, λ: 1e-05	max	57.3792
SR	TRUE	η: 0.1, Niter: 1000, λ: 0.0001	min	55.0442
SR	FALSE	η: 0.1, Niter: 1000, λ: 1e-06	product	56.521
SR	FALSE	η: 0.1, Niter: 1000, λ: 0.0001	sum	56.4584
SR	FALSE	η: 0.1, Niter: 1000, λ: 0.0001	vote	56.0154
SR	**FALSE**	**η: 1, Niter: 500, λ: 0.0001**	**clf**	**61.2447**

Unlike the Twittter-sanders-apple2 dataset, however, all other rules degrade the performance of the base classifier, including the 'vote' rule that was able to improve the base model in the Twittter-sanders-apple2 dataset. The improvement in the best model is also lower than the Twittter-sanders-apple2 dataset. The SR ensemble gained a 3% improvement from the base model on the Twittter-sanders-apple2 dataset, whereas it is only 0.7% improvement on the Twittter-sanders-apple3 dataset.

The third result is Testdata-manual dataset in Table 8. This is a fairly balanced dataset with only IR of 1.30, based on Table 2. The results for the Testdata-manual dataset are presented in Table 8. The result shows that ensemble models generally degrade the performance of the base classifier, regardless of the rule. The only exception is the SR model. The SplitBal scheme split the majority class into multiple partitions to balance the majority and minority classes [14]. However, when the dataset is more balanced like the Testdata-manual dataset, this will lead to fewer majority class samples in each classifier. As the sample decreases, the generalization ability also degrades. In the case of the Testdata-manual dataset, the majority class 'positive' is divided from 180 samples into 90 samples for each classifier. Trends from Twitter-sanders-apple2 and Twitter-sanders-apple3 datasets still apply to this dataset. NB models perform worse than SGD models. SR once again is the best model. The parameter of SR is also the same, i.e. learning rate $\eta = 1$, number of iterations $Niter = 500$, and regularization parameter $\lambda = 0.0001$. Ensembled SR is also even able to improve the performance of base SR although the improvement is only 1%.

Table 8. Classification Results for Testdata-manual Dataset

classifier	scaling	parameter	rule	f1-macro
MNB	***FALSE***	**-**	**-**	***62.2455***
MNB	FALSE	-	max	60.5493
MNB	FALSE	-	min	59.5383
MNB	FALSE	-	product	60.297
MNB	FALSE	-	sum	60.2881
MNB	FALSE	-	vote	55.5382
MNB	FALSE	-	clf	62.1199
CNB	***FALSE***	**-**	**-**	***62.8633***
CNB	FALSE	-	max	60.7403
CNB	FALSE	-	min	60.639
CNB	TRUE	-	product	61.0123
CNB	TRUE	-	sum	60.7535
CNB	FALSE	-	vote	56.4198
CNB	FALSE	-	clf	61.4414
SVM	***FALSE***	***η: 0.1, Niter: 500, λ: 1e-06***	**-**	***65.6313***
SVM	TRUE	η: 0.1, Niter: 500, λ: 0.0001	max	64.1782
SVM	TRUE	η: 0.1, Niter: 1000, λ: 0.0001	min	62.5893
SVM	TRUE	η: 0.1, Niter: 1000, λ: 0.0001	product	63.8692
SVM	TRUE	η: 0.1, Niter: 500, λ: 1e-05	sum	64.1523
SVM	FALSE	η: 0.001, Niter: 1000, λ: 1e-05	vote	58.571
SVM	FALSE	η: 0.1, Niter: 1000, λ: 1e-06	clf	65.4368
SR	*TRUE*	*η: 1, Niter: 2000, λ: 0.0001*	-	*65.1994*
SR	TRUE	η: 1, Niter: 500, λ: 1e-06	max	64.7618
SR	FALSE	η: 1, Niter: 1000, λ: 1e-05	min	63.3023
SR	TRUE	η: 1, Niter: 1000, λ: 1e-05	product	63.693
SR	TRUE	η: 1, Niter: 1000, λ: 1e-06	sum	64.1649
SR	FALSE	η: 0.01, Niter: 2000, λ: 0.0001	vote	59.2455
SR	**TRUE**	**η: 1, Niter: 500, λ: 0.0001**	**clf**	**66.1684**

For the Airline dataset, the result is presented in Table 9. The Airline dataset has an IR of 3.89, which means that the majority class samples are adequately distributed into each ensemble member. Even though the Airline dataset has approximately 15 times more data than, the result shows a similar trend to the Twitter-sanders-apple3 dataset, with.IR of 3.14. In the Airline dataset, all base methods are improved by the ensemble with 'clf' rule, although other rules also decrease the performance. The preferred model is still the SR model, followed by SVM, and then by the two NB models. The improvement is also lower. MNB and SVM gained 2% improvement, while CNB and SR models gained 1% or less improvement.

The last experiment is the Self-driving dataset with highly-imbalanced distribution with IR of 38.59. The results are presented in Table 10. The minority class only has 110 samples, compared to 4245 samples of the majority class. This dataset has the lowest F1-macro of all datasets. Also, almost all base classifier models have higher performance than the ensembled models. The only exception is the ensembled MNB model which performed better than the base MNB model. However, the ensembled MNB model is also comparatively worse than other base classifier models. As Self-driving is highly imbalanced, the number of ensemble members is also large [14]. This can lead to a sub-optimal combination of decisions. The probability of a worse-performing classifier affecting the overall decision is also higher.

Table 9. Classification Results for Airline Dataset

classifier	scaling	parameter	rule	f1-macro
MNB	*FALSE*	-	-	*64.2757*
MNB	FALSE	-	max	65.3538
MNB	FALSE	-	min	64.8329
MNB	FALSE	-	product	65.2236
MNB	FALSE	-	sum	65.5195
MNB	FALSE	-	vote	63.7973
MNB	**FALSE**	**-**	**clf**	**66.7791**
CNB	*FALSE*	-	-	*64.7696*
CNB	FALSE	-	max	64.8327
CNB	FALSE	-	min	64.5059
CNB	FALSE	-	product	65.1154
CNB	FALSE	-	sum	65.1397
CNB	FALSE	-	vote	63.856
CNB	**FALSE**	**-**	**clf**	**65.2419**
SVM	*TRUE*	*η: 1, Niter: 500, λ: 0.0001*	-	*66.5517*
SVM	TRUE	η: 0.1, Niter: 2000, λ: 0.0001	max	65.6016
SVM	FALSE	η: 0.1, Niter: 1000, λ: 1e-07	min	64.9473
SVM	FALSE	η: 0.1, Niter: 2000, λ: 0.0001	product	66.1497
SVM	FALSE	η: 0.1, Niter: 2000, λ: 1e-06	sum	66.2155
SVM	TRUE	η: 0.1, Niter: 1000, λ: 0.0001	vote	65.1762
SVM	**TRUE**	**η: 0.1, Niter: 2000, λ: 0.0001**	**clf**	**67.5549**
SR	*TRUE*	*η: 1, Niter: 2000, λ: 0.0001*	-	*66.827*
SR	TRUE	η: 1, Niter: 2000, λ: 0.0001	max	65.9654
SR	FALSE	η: 1, Niter: 2000, λ: 0.0001	min	65.1886
SR	FALSE	η: 1, Niter: 1000, λ: 0.0001	product	66.1923
SR	FALSE	η: 1, Niter: 1000, λ: 0.0001	sum	66.1985
SR	TRUE	η: 1, Niter: 1000, λ: 0.0001	vote	65.3042
SR	**FALSE**	**η: 1, Niter: 2000, λ: 0.0001**	**clf**	**67.6953**

Based on classification results from five datasets, the ensemble of the SR model is the overall best classifier. Ensembled SR with the 'clf' rule can score the highest in four of five datasets, while the one dataset is best classified with base SR model. All SR models also have learning rate $\eta = 1$, with a varying number of iterations and regularization.

4.2 Running Time

After comparing classification results, the training time is compared. The average training time for each model is presented in Fig. 5. The training time of the ensemble depends on the number of classifiers and the type of the classifier itself. NB models only require one pass of the dataset to train the model, and therefore only have training time in the order milliseconds. Even when training multiple classifiers, the NB ensemble is comparatively much faster to train compared to SVM/SR ensembles.

For SGD models (SVM/SR), the training time primarily depends on the number of iterations. As can be seen in Fig. 5, more iterations lead to longer training time. The training time is also dependent on the number of classifiers in the ensemble. The number of classifiers in the SplitBal method equals the imbalance rate (IR) of the dataset [14] e.g., a dataset with IR of 4 will be trained using an ensemble of 4 classifiers. The training time, however, depends less on the size of the dataset. The use of the sparse data format ensures that training time is less dependent on the number of training samples. For example, the Airline dataset has 29 times more data than Twitter-sanders-apple2 but only takes 3 times longer to train a 2000 iterations SR ensemble with twice more classifiers.

Table 10. Classification Results for Self-driving Dataset

classifier	scaling	parameter	rule	f1-macro
MNB	*FALSE*	-	-	*26.9179*
MNB	FALSE	-	max	29.0699
MNB	TRUE	-	min	15.8256
MNB	TRUE	-	product	16.8531
MNB	FALSE	-	sum	18.7872
MNB	FALSE	-	vote	20.3371
MNB	**TRUE**	**-**	**clf**	**30.7036**
CNB	***FALSE***	**-**	**-**	***31.4554***
CNB	TRUE	-	max	27.8036
CNB	FALSE	-	min	21.5052
CNB	TRUE	-	product	21.6114
CNB	FALSE	-	sum	22.453
CNB	TRUE	-	vote	22.6131
CNB	FALSE	-	clf	31.0813
SVM	***TRUE***	***η: 0.1, Niter: 2000, λ: 1e-07***	**-**	***34.2339***
SVM	TRUE	η: 0.1, Niter: 500, λ: 0.0001	max	29.7208
SVM	TRUE	η: 0.01, Niter: 1000, λ: 0.0001	min	18.6483
SVM	TRUE	η: 0.001, Niter: 1000, λ: 0.0001	product	23.8085
SVM	TRUE	η: 0.001, Niter: 1000, λ: 1e-07	sum	24.1977
SVM	TRUE	η: 0.01, Niter: 1000, λ: 1e-05	vote	23.3882
SVM	TRUE	η: 0.1, Niter: 2000, λ: 0.0001	clf	32.879
SR	***TRUE***	***η: 1, Niter: 2000, λ: 1e-05***	**-**	***34.441***
SR	TRUE	η: 1, Niter: 500, λ: 1e-05	max	30.13
SR	TRUE	η: 0.1, Niter: 500, λ: 1e-05	min	19.1068
SR	TRUE	η: 0.01, Niter: 500, λ: 0.0001	product	22.9776
SR	TRUE	η: 0.01, Niter: 500, λ: 1e-05	sum	23.3909
SR	TRUE	η: 0.01, Niter: 1000, λ: 1e-05	vote	23.0567
SR	TRUE	η: 1, Niter: 1000, λ: 0.0001	clf	33.4769

Although training does not take more than 10 seconds on the most dataset, the cost of the training SVM/SR ensembles is the most noticeable on self-driving because of the IR of 39. It takes almost 60 seconds to train the whole ensemble. This means that 4-fold cross-validation will last for 4 minutes to test a single parameter combination. This leads to a very long time to find optimal parameters. Additionally, the best ensemble model does not outperform the base model, and performing a grid search on the base model is much faster. Testing time is not reported in this research. Testing unknown data samples with ensemble only takes time in the order of milliseconds. Unlike training process that involves iterative process (for SVM/SR), testing only involves one non-iterative process. As mentioned, the use of sparse data format ensures efficiency. This also means that the testing time for each rule is not compared, as the ensemble can use all rule without re-training.

5. Conclusions

In this research, a comparison of the ensemble model to find the most suitable method for classifying various datasets is performed. The ensemble models compared are 'max', 'min', 'product', 'vote', 'sum', and 'clf' models. Four base classifiers, Multinomial NB, Complement NB, Support Vector Machine, and Softmax Regression are compared to see the effect on the resulting ensemble model. The ensembles models are then tested on five datasets with different characteristics in the number of data and the imbalanced ratio.

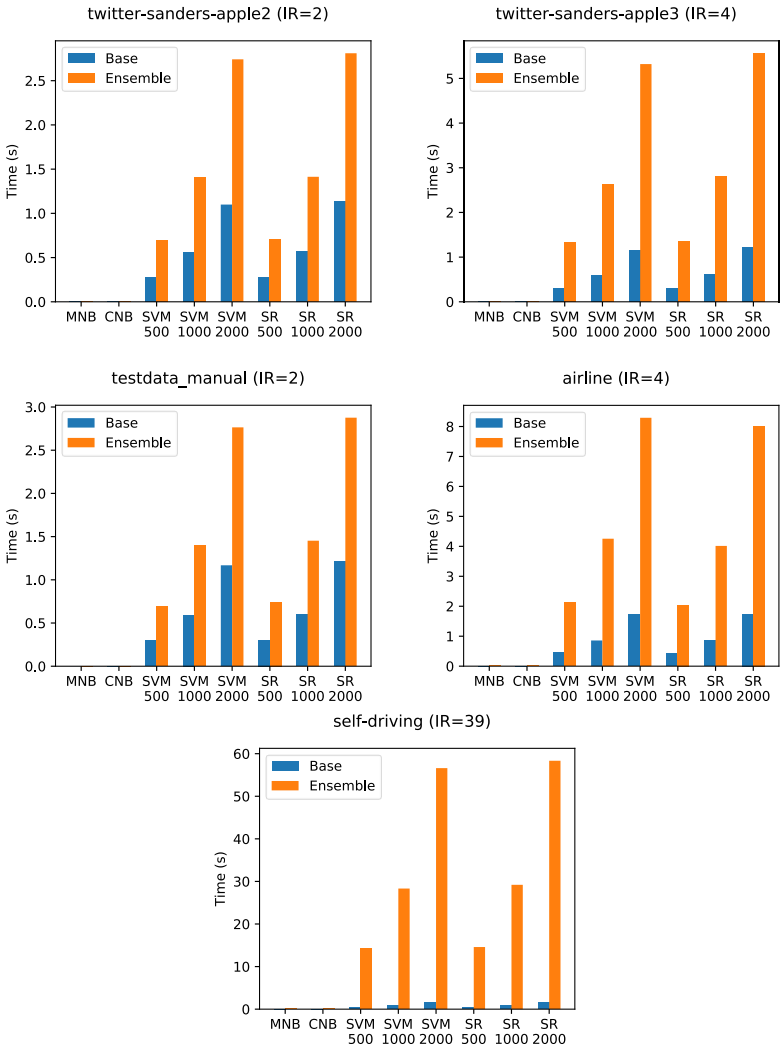

Fig. 5 Training Time Comparison for Each Dataset

From experimental results, it is found that 'clf' (using meta-classifier to combine the results of the ensemble members) rule is the most consistent in improving the base classifier. 'max' and 'vote' rules can improve the base classifier in some datasets. However, other rules degrade the base classifier in most cases. As the classifier of choice, Softmax Regression is the best performing classifier in all datasets. Ensembled Softmax classifier is also the best model in all but one dataset. From this, it can be concluded that the ensemble model with the 'clf' rule can improve the performance of the base classifier.

For future researches, datasets with highly-imbalanced data such as self-driving can be observed. Additionally, the ensemble method that assigns weight to each classifier can be used. There are several ways to perform weighting, such as evolutionary algorithm-based weighting. There is also the need to build an efficient ensemble, as it can be very time consuming to perform an ensemble of certain classifiers such as Support Vector Machine and Softmax Regression.

References

1. M. S. Akhtar, D. Gupta, A. Ekbal, and P. Bhattacharyya, "Feature selection and ensemble construction: A two-step method for aspect based sentiment analysis," *Knowledge-Based Syst.*, vol. 125, pp. 116–135, 2017.
2. B. Liu, "Sentiment analysis and opinion mining," *Synth. Lect. Hum. Lang. Technol.*, vol. 5, no. 1, pp. 1–167, 2012.
3. Y. Zhang *et al.*, "Does deep learning help topic extraction? A kernel k-means clustering method with word embedding," *J. Informetr.*, vol. 12, no. 4, pp. 1099–1117, 2018.
4. G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, "Learning from class-imbalanced data: Review of methods and applications," *Expert Syst. Appl.*, vol. 73, pp. 220–239, 2017.
5. L. Muflikhah and D. J. Haryanto, "High performance of polynomial kernel at SVM Algorithm for sentiment analysis," *J. Inf. Technol. Comput. Sci.*, vol. 3, no. 2, pp. 194–201, 2018.
6. M. Z. Sarwani and W. F. Mahmudy, "Campus Sentiment Analysis E-Complaint Using Probabilistic Neural Network Algorithm," *J. Ilm. Kursor*, vol. 8, no. 3, 2016.
7. I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, no. Mar, pp. 1157–1182, 2003.
8. N. Burns, Y. Bi, H. Wang, and T. Anderson, "Sentiment Analysis of Customer Reviews: Balanced versus Unbalanced Datasets BT - Knowledge-Based and Intelligent Information and Engineering Systems," 2011, pp. 161–170.
9. L. Yijing, G. Haixiang, L. Xiao, L. Yanan, and L. Jinling, "Adapted ensemble classification algorithm based on multiple classifier system and feature selection for classifying multi-class imbalanced data," *Knowledge-Based Syst.*, vol. 94, pp. 88–104, 2016.
10. N. V Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, 2002.
11. M. A. Tahir, J. Kittler, K. Mikolajczyk, and F. Yan, "A multiple expert approach to the class imbalance problem using inverse random under sampling," in *International Workshop on Multiple Classifier Systems*, 2009, pp. 82–91.
12. S. H. Khan, M. Hayat, M. Bennamoun, F. A. Sohel, and R. Togneri, "Cost-sensitive learning of deep feature representations from imbalanced data," *IEEE Trans. neural networks Learn. Syst.*, vol. 29, no. 8, pp. 3573–3587, 2017.
13. R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits Syst. Mag.*, vol. 6, no. 3, pp. 21–45, 2006.
14. Z. Sun, Q. Song, X. Zhu, H. Sun, B. Xu, and Y. Zhou, "A novel ensemble method for classifying imbalanced data," *Pattern Recognit.*, vol. 48, no. 5, pp. 1623–1637, 2015.
15. L. Jiang, C. Li, S. Wang, and L. Zhang, "Deep feature weighting for naive Bayes and its application to text classification," *Eng. Appl. Artif. Intell.*, vol. 52, pp. 26–39, 2016.
16. D. M. Diab and K. M. El Hindi, "Using differential evolution for fine tuning naïve Bayesian classifiers and its application for text classification," *Appl. Soft Comput.*, vol. 54, pp. 183–199, 2017.
17. L. Jiang, L. Zhang, C. Li, and J. Wu, "A correlation-based feature weighting filter for Naive Bayes," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 2, pp. 201–213, 2019.
18. R. A. Cahya and F. A. Bachtiar, "Weakening Feature Independence of Naïve Bayes Using Feature Weighting and Selection on Imbalanced Customer Review Data," in *2019 5th International Conference on Science in Information Technology (ICSITech)*, 2019, pp. 182–187.
19. M. N. Injadat, F. Salo, and A. B. Nassif, "Data mining techniques in social media: A survey," *Neurocomputing*, vol. 214, pp. 654–670, 2016.
20. M. Jiang *et al.*, "Text classification based on deep belief network and softmax regression," *Neural Comput. Appl.*, vol. 29, no. 1, pp. 61–70, 2018.
21. Q. Jiang, W. Wang, X. Han, S. Zhang, X. Wang, and C. Wang, "Deep Feature Weighting In Naive Bayes For Chinese Text Classification," in *Proceedings of CCIS2016*, 2016, pp. 1–5.
22. A. C. Pandey, D. S. Rajpoot, and M. Saraswat, "Twitter sentiment analysis using hybrid

cuckoo search method," *Inf. Process. Manag.*, vol. 53, no. 4, pp. 764–779, Jul. 2017.
23. L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, 1996.
24. D. H. Wolpert, "Stacked generalization," *Neural networks*, vol. 5, no. 2, pp. 241–259, 1992.
25. J. D. Rennie, L. Shih, J. Teevan, and D. R. Karger, "Tackling the poor assumptions of naive bayes text classifiers," in *Proceedings of the 20th international conference on machine learning (ICML-03)*, 2003, pp. 616–623.
26. L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*, Springer, 2010, pp. 177–186.
27. T. Liu, "A novel text classification approach based on deep belief network," in *International Conference on Neural Information Processing*, 2010, pp. 314–321.
28. B. Zadrozny and C. Elkan, "Transforming classifier scores into accurate multiclass probability estimates," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002, pp. 694–699.
29. D. Agnihotri, K. Verma, and P. Tripathi, "Variable Global Feature Selection Scheme for automatic classification of text documents," *Expert Syst. Appl.*, vol. 81, pp. 268–281, 2017.