Applied Soft Computing Journal 97 (2020) 106770

Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

TB-CoAuth: Text based continuous authentication for detecting compromised accounts in social networks

Ravneet Kaur [*], Sarbjeet Singh, Harish Kumar

University Institute of Engineering and Technology, Panjab University, Chandigarh, India

ARTICLE INFO

Article history:
Received 18 November 2019
Received in revised form 25 September 2020
Accepted 28 September 2020
Available online 6 October 2020

Keywords:
Authorship verification
Continuous authentication
Compromised accounts
Classification
Social networks

ABSTRACT

Commencement of research towards compromised account detection in email and web services foreshadows the growth of the same in social network scenario. In this paper, continuous authentication of textual content has been performed for incessant authorship verification to detect compromised accounts in social networks. Four categories of features namely, content free, content specific, stylometric and folksonomy are extracted and evaluated. Experiments are performed with 3057 twitter users taking 4000 latest tweets for each user. It is evident from the experiments that consistency maintained on features is different for each user. Hence, various statistical and similarity-based feature selection techniques are used to rank and select optimal features for each user which are further combined using a popular rank aggregation technique called BORDA. Also, performance of various supervised machine learning classifiers is analyzed on the basis of different evaluation metrics. Experimental results state that for the undertaken problem, SVM with *rbf* kernel outperformed other classifiers namely, kNN, Random Forest, Gradient Boosted and Multi Layer Perceptron, attaining a maximum F-score of 94.57% under the varied parameter settings.

1. Introduction

Social networks have become an integral part of the daily routine of people throughout the world and people spend a vast amount of their time on these platforms. It is evident from the 2017 statistics that internet users nearly spend 135 min on any social network in a day.[1] But with the increasing popularity of social networks, it is not uncommon to hear the frequent breaches and impersonation attacks on these platforms. In 2016, a Pew Research study was conducted in which it was found that around 13% of adults experienced compromisation of their online social media accounts [1].

Also, as per Grier et al. [2], 84% of the spam accounts on Twitter were compromised while only 16% of them were fake. Similarly, on Facebook also, 97% of malicious accounts spreading spam belonged to genuine users [3]. Adding on to this, another research states that 10% of the compromised victims believe that they were hacked by a friend or an acquaintance [4] and 16% of the victims easily identified their hacker by name or relation [5]. Above statistics marks the severity and impact of compromised accounts in social networks.

Albeit the severity and proliferation of accounts getting compromised on various social networks, work on detecting compromised accounts in social networks is still in its infancy. Till now academicians and practitioners are only concerned with the development of remedial measures to detect and deal with spam and fake accounts. Only a few theoretical approaches have been developed to deal with this sensitive problem of compromised account detection which is otherwise a topic of huge concern. Even service providers of popular social networks rely on point-of-entry based approaches such as IP geolocation based authentication at the time of login which seems to be insufficient. After the legitimate login into an account, the authenticated session can be easily commandeered by an intruder through various session hijacking attacks. Hence, login authentication should be supplemented with other reliable practices that could be applied unobtrusively at the back end. One such practice is the use of continuous authentication (CA) to re-verify the user during the session. Continuous Authentication of data entails good accuracy with small authentication delay as well as confrontation to manipulation and forgery. It can be applied either at the front end in the form of biometric authentication or unobtrusively at the back end at stylometric level. It is evident from the literature that use of biometric modalities for continuous authentication has been in wide practice since long but use of stylometry based CA is still in its embryonic state with only a few works primarily dedicated towards it [6–10].

* Corresponding author.
E-mail addresses: ravneets48@gmail.com (R. Kaur), sarbjeet@pu.ac.in (S. Singh), harishk@pu.ac.in (H. Kumar).

1 http://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/

https://doi.org/10.1016/j.asoc.2020.106770
1568-4946/© 2020 Elsevier B.V. All rights reserved.

Table 1
Prominent authorship verification tasks in social networks.

Ref.	Social network	Dataset statistics	Features	Feature selection	Approach	Varied parameters	Evaluation metrics	Remarks
[7,11]	Twitter and Enron Emails	**Twitter:** 100 users (Avg 3194 tweets) **Enron:** 150 users (76 after preprocessing) (Avg 0.2M emails)	Lexical, Syntactic, Application Specific, n-grams	Information Gain (discarding features based on Mutual Information)	SVM, Hybrid SVM-LR Classifier	Block size, Number of blocks, feature selection, weight of negative samples, SVM-kernel	FAR, FRR, EER	EER ranging from 9.98% to 21.45%(for different block sizes)
[9]	Twitter and Enron	**Enron:** 150 users (Avg. 757 messages per user) **Twitter:** 100 users (Avg. 3194 tweets) **Impostors:** 10 users impersonating authors	Lexical, Syntactic, Application Specific	Information Gain and Mutual Inclusion	Deep Belief Networks (GB-DBN and RBMs)	Block Sizes, hidden layers, epochs	FRR, FAR, EER, HTER	**Twitter:** EER: 11.48% (BB-RBM), 10.08% (GB-RBM) **Email:** FRR: 8.24%, FAR: 8.2%, EER: 8.21%, HTER: 8.22% **Imposters:** EER: 5.48% (280 char BS), 12.3% (140 char BS), HTER: 6.68%
[12]	Facebook	30 users (9259 Facebook posts)	Stylometric and social network features	Manual Selection	kNN, SVM, Naive Bayes, Decision Tree, Neural Network	Features, users, normalization, classifiers	Accuracy	**Accuracy:** 79.6% Small data set — hence, LOO was applied for cross- validation
[13]	Website comments	40 authors	Bit level n-grams	Manual Selection	kNN	n (bits) in n-grams, k in kNN classification	Accuracy, FRR, FAR	Avg. accuracy: 90% Outlier: Avg. accuracy: above 80%
[14]	Twitter	1000 users (at most 500 tweets for each user)	Char n-grams (SPI Similarity measure)	–	kNN	Interspersing factor λ, n in n-grams, k in kNN classifier	Accuracy, Precision, FRR, TNR, ROC	**Accuracy:** 93% **FNR:** 11%–12% **TNR:** 99% (approx.)
[5]	Twitter	5889 users — after filtering 5527 (462 compromised, 5065-normal)	Content, source, location, timing	Manual selection	CADET, PCA, NMF, COMPA	Logistic Regression (LR)	F-measure	F-measure: 0.41
[15]	Twitter	1000 users (at most 500 tweets for each user)	n-grams (char, word), BOW(SPATIUM-L1, tf-idf), Topic Modeling, Folksonomy	AHP-TOPSIS, Fisher Score, t-score, Chi-square, Correlation Feature selection, Gini Index	kNN	k in kNN classifier, r for Interspersing	Accuracy, Precision, Recall, Fscore, FRR, FAR, CV	**Average accuracy:** 93.48% **Fscore:** 93.82% **FAR:** 7.15% **FRR:** 5.88%
[10]	Enron and Twitter	**Enron:** 158 users (Avg. 757 messages per user) **Twitter:** 100 users (Avg. 3194 tweets) **Impostors:** 10 users	Lexical, Syntactic and application specific	Information Gain and Mutual Inclusion	Shallow Classifiers: LR, SVM, LR-SVM Deep Learning: Deep Belief Network (DBN)	Block sizes, Blocks per user	EER	Best performance: **Enron:** EER: 9.18% (LR) **Twitter:** EER: 11.83% (LR and BS: 280), 10.08% (BS: 280 and 100 blocks/user) **Impostors:** EER: 5.48% (BS: 280)
[16]	Twitter	5065 normal and 462 compromised users (200 tweets of each user were extracted)	Content, Source, Location and Timing	Non-linear transformation of features using sigmoid activation function	Non linear autoencoder	K value for Top k%,	P@K, Reconstruction Error, AU-PR curve	• Proposed multi-view framework • Considers each data modality as a view • Nonlinear autoencoders learn feature embeddings from multiple views • For each view, the data is encoded into a lower dimension feature space and then reconstructed (decoded) into its original representation
[17]	Twitter	7632 normal and 835 compromised users	Temporal, Context and Network	–	LSTM (ReLU activation function), word embeddings, Clustering	Learning rate, hyperparameter tuning	Precision, Recall, F-score, AUC	• Temporal correlations using LSTM • Sparsity problem using contextual information (doc2vec representation) • Connectivity using network features (Local Clustering Co-efficient) • F-score of 0.33 (with the combination of temporal, context and network features)

Abbreviations: BOW: Bag of Words, FAR: False Acceptance Rate, TNR: True Negative Rate, FRR: False Rejection Rate, EER: Equal Error Rate, CV: Coefficient of Variance, LOO: Leave One Out, kNN: k Nearest Neighbor, LR: Logistic Regression, SVM: Support Vector Machine, BS: Block Size.

Major objective of this research is to perform text based continuous authentication and investigate whether profiling user's textual and stylometric behavior could help detect the authenticity of an incoming message from a user profile. Furthermore, this paper focuses on the viability of authorship verification process for the detection of compromised accounts and accordingly the problem has been structured as a document representation model. Unlike existing works, this paper examines the task of authorship verification on short texts and applies text mining tasks for stylometric analysis of short excerpts of social media text. Also, efficiency of various machine learning classifiers have been analyzed in order to yield an optimal classifier to be considered for the automated process.

Following research aspects have been covered in this paper:

1. Analysis of the efficiency of various textual features for continuous authentication and identification of compromised accounts in social networks.
2. Selecting optimal features for each individual user for further classification.
3. Examining the viability of supervised machine learning classification approaches for continuous authentication of online social network content.

Rest of the paper is structured as follows. Section 2 reviews some of the works relevant to the authorship verification of online content with the focus on works that made use of text based features. Section 3 discusses the insights drawn from the related literature and accordingly the problem formulation. Section 4 covers the proposed framework, features and classification approaches adopted for the verification and detection of compromised accounts followed by the demonstration of experimental results in Section 5. Finally, Section 6 concludes the paper discussing some future insights into the work.

2. Related work

Authorship analysis involves the exploration of textual excerpts to determine the authorship and has been the key area of research for years. Considering the distribution of such tasks into three major categories [18] namely, attribution [6,19,20], verification [7,14,21] and characterization [22,23], the process of compromised account detection can be best correlated to authorship verification which involves recognizing whether the text in question is written by the acclaimed user or not. What motivates the current research is the increasing inclination of automatic inference of authenticity of the text in question. This automatic authenticity has been studied for different purposes, ranging from literary works [24,25] to plagiarism detection [26,27], forensic investigations [21,28,29] and cyber security problems [14,15,30]. In a similar note, this research work also examine the applicability of the authorship verification task for the detection of compromised accounts. In order to build a strong foundation, this section discuss some of the existing works already in place towards the authorship verification of social network messages. Table 1 gives a description of data statistics and few important parameters extracted from some of the relevant previous works in the scientific literature. Discussed parameters namely, the social network used, type and count of data, features deployed, classifiers and evaluation metrics used are the key attributes looked in any work. Choice of such parameters has always been a debatable issue but till now no specific parameters have been standardized.

From textual perspective, authorship verification problems are mainly handled using content-specific or content-free analysis. Secondly, most of the related works have seen the deployment of machine learning approaches to handle the problem. Also, it has been analyzed that researchers have progressively worked towards the domain beginning with some preliminary analysis and then improving the work by incorporating various additional factors. As an example, **Brocardo and Traore** [11] initially performed a preliminary work for the authorship verification of short length Enron e-mails using n-grams as features obtaining an Equal Error Rate, EER (a point with same values for False Acceptance Rate (FAR) and False Rejection Rate (FRR)) of 14%. Further, time and again continuous authentication experiments were performed by the same researchers on different corpus using various textual features and classification approaches. As an extension to the above stated work [11], experiments were performed by **Brocardo** et al. [7] on e-mail and Twitter data using lexical, syntactic and application centered textual features. A hybrid classifier was built by integrating the efficiency of Logistic Regression (LR) and SVM in a manner that the output from SVM was input to the logistic function of LR. Later, **Brocardo** et al. [9] themselves examined the glitch of machine learning architectures and approaches for the authorship verification task and analyzed how deep belief networks attained more promising results (an error rate of 5 to 12%). GaussianBernoulli deep-belief network (GB-DBN) was used by researchers with a layered framework of Restricted Boltzmann Machines (RBMs) followed by a supervised machine learning classifier for classification. Alongside syntactic, lexical, and application-specific features, various combination of n-grams were also analyzed. Again as an extension to this work, in [10] experiments were performed with various shallow and deep learning classifiers to investigate the use of various stylometric features.

Most of the other studies documented in the literature have also relied on the text mining taking n-grams as features such as, **Igawa** et al. [31] and **Barbon** et al. [14] proposed a text mining based technique using only char n-grams as profile characteristics. A similarity measure called Simplified Profile Intersection (SPI) was used to verify the authenticity of tweets on the social networking platform, Twitter. SPI values of positive and negative test sets were evaluated using an instance based classifier (kNN) and a FIFO policy was adopted to dynamically update the baseline profile which helped to improve accuracy and reduce outliers. Unlike existing works that considered n-grams as byte based **Peng, Choo, & Ashman** [13] used bit-level n-grams for authorship attribution and verification of online social media data. Topic independent data along with informal language data such as emoticons and abbreviations were also analyzed.

Some of the other widely used approaches made use of content-free attributes such as stylometric features. For instance, **Li** et al. [12] examined the efficiency of n-grams as well as different stylometric and social network features for short length messages. Alongside, a comparative analysis of performance of different machine learning classifiers namely, k-nearest neighbor, Support Vector Machine, Naive Bayes, Decision trees and Neural Network was done for authorship verification of corresponding posts on Facebook. A voting algorithm was used to combine the output of various classifiers. Though, an increase in the count of features helped to improve the accuracy but only 30 users were taken for experiments which was a very poor sample in comparison to the millions of active users worldwide.

Kaur et al. [15] also analyzed the competence of various textual features for the authorship verification of social network posts. Alongside various traditional textual features some stylometric, folksonomic and topic modeling features were also used. It was inferred that users did not stay consistent on same set of features, hence, AHP-TOPSIS was used to rank features for each respective user. The problem was studied as a classification problem deploying kNN classifier for the task. Though, an in depth analysis for each user was performed but there remains

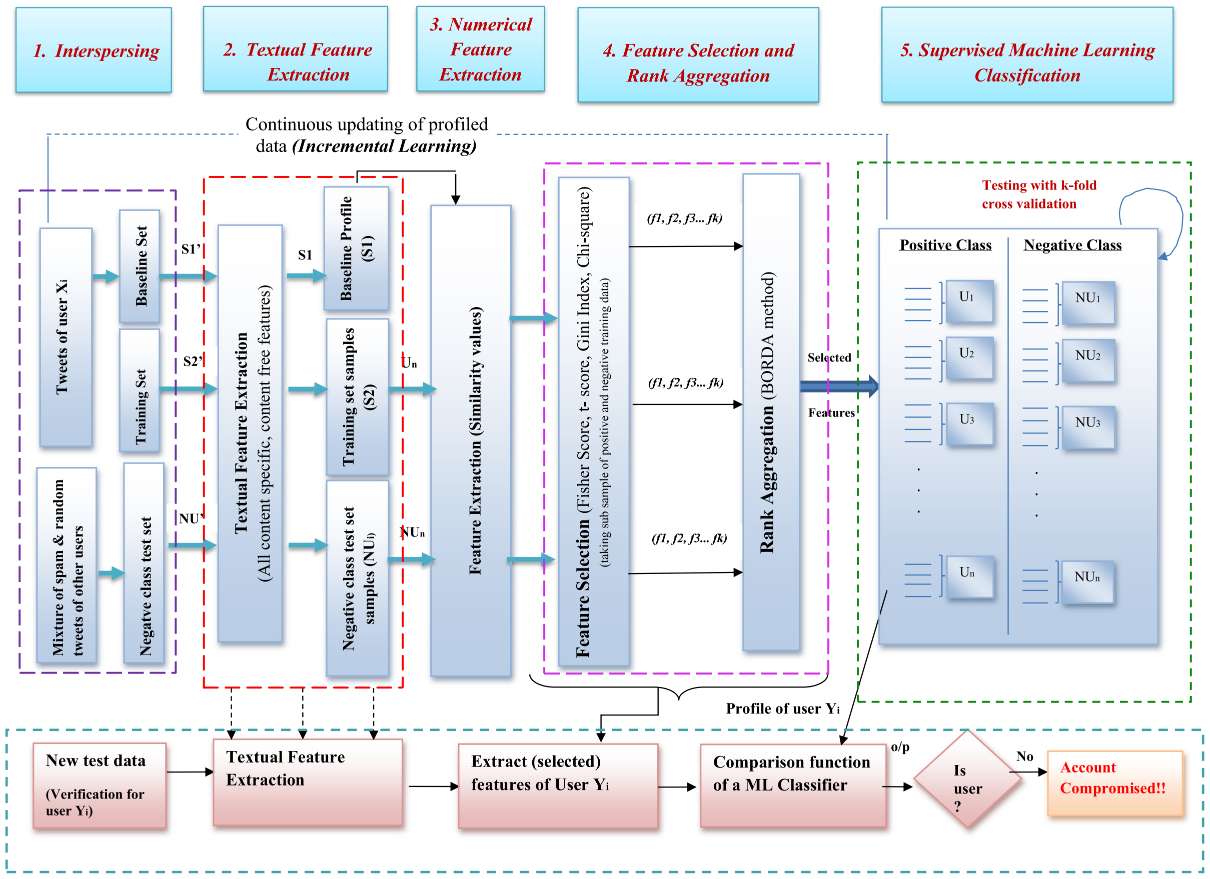

1. Interspersing
2. Textual Feature Extraction
3. Numerical Feature Extraction
4. Feature Selection and Rank Aggregation
5. Supervised Machine Learning Classification
Continuous updating of profiled data (Incremental Learning)
Tweets of user Xi
Baseline Set
Training Set
Mixture of spam & random tweets of other users
Negatve class test set
S1'
S2'
NU'
Textual Feature Extraction (All content specific, content free features)
S1
Baseline Profile (S1)
Training set samples (S2)
Negative class test set samples (NUi)
Un
NUn
Feature Extraction (Similarity values)
Feature Selection (Fisher Score, t- score, Gini Index, Chi-square) (taking sub sample of positive and negative training data)
(f1, f2, f3... fk)
Rank Aggregation (BORDA method)
Selected Features
Testing with k-fold cross validation
Positive Class
Negative Class
U1 U2 U3 Un
NU1 NU2 NU3 NUn
Profile of user Yi
New test data (Verification for user Yi)
Textual Feature Extraction
Extract (selected) features of User Yi
Comparison function of a ML Classifier
o/p
Is user ?
No
Account Compromised!!

Fig. 1. Proposed TB-CoAuth approach for continuous authentication and compromised account detection.

a need to carry out the experiments with some more classification techniques, additional users as well as more train and test samples.

Some of the recent research works have also seen the deployment of unsupervised methods to detect or analyze the compromised accounts. As an example, unsupervised modeling of normal behavior using PCA helped **Viswanath** et al. [32] in efficient detection of anomalous changes in behavioral pattern. Proposed approach helped to identify 'like' and 'click' spammers that exploited social networks using the creation of fake, compromised or other colluding accounts.

Likewise, **VanDam** et al. [16] proposed an unsupervised multi-view learning framework, CADET, utilizing the nonlinear autoencoders for feature embedding from different views like, content of tweets, location, source and timing. Proposed framework targeted the user-level detection of compromised accounts. For performance evaluation, all users were ranked according to their reconstruction errors. Further, top-k% reported compromised users were selected to be evaluated (by domain experts) on the basis of P@K (Precision at top-K) measures.

Karimi et al. [17] presented an End-to-End Compromised Account Detection Framework named E2ECAD to detect compromised accounts in social networks by taking contextual, temporal and network information into consideration. Firstly, E2ECAD handles the problem of sparse social network data by incorporating the contextual information using embeddings (word embeddings and doc2vec). Secondly, temporal correlations among activities were handled using LSTM networks. Thirdly, network features were also considered to cover the user connectivity aspect. Performance obtained by E2ECAD is quite low (F-score of 0.33) for its deployment in actual real time scenario.

3. Insights drawn from the literature and problem statement

It is seen that in respect to the approach adopted to handle the authorship verification problem, majority of the studies have aimed at using either a threshold-based approach [14,33–37] or deployment of machine learning classification models [12,14, 15,31]. Threshold based methods however, limits the model for flexibility and dynamic updates. But with the advent of machine learning in almost every domain, applicability of different machine learning approaches for the authorship verification task is also evident. As per the literature in place, it is seen that a supervised or unsupervised machine learning approach have been adopted to solve the problem of compromised account detection. Usage of these techniques help learn the patterns and automate the process of authentication. From textual perspective, most of the existing techniques have relied on the use of different supervised learning approaches. Only a handful of studies focused on the deployment of unsupervised learning methods for the task [5,17,38]. Also it is observed that not only the performance efficiency obtained by such techniques was less than that obtained by the supervised methods, but also such techniques needed a final manual evaluation by domain experts. Moreover, based on the problem domain in consideration and the general approach used to follow it, applicability of supervised learning classifiers seems more viable for the task. It is more evident from the following problem statement which defines the way the problem of compromised account detection is handled in this work:

"Given a social network account X, its temporally sequential posts P and the corresponding binary labels Y, we aim to learn the model M for X mapping P to Y, which can automatically predict the labels for unseen account posts".

Table 2
Textual Features and the corresponding similarity metric.

Count	Feature	Remarks	Similarity feature metric	Category
2	Char n-grams	*n* characters are considered as a unit	Simplified Profile Intersection (SPI)	R0
			Jaccard Coefficient (JC)	R1
2	Unigram	Single word is taken as a unit	Simplified Profile Intersection (SPI)	R2
			Jaccard Coefficient (JC)	R3
2	Bigram	Two words are taken as a unit	Simplified Profile Intersection (SPI)	R4
			Jaccard Coefficient (JC)	R5
1	token prefix	*k* starting letters of each token are considered as a unit	Simplified Profile Intersection (SPI)	R6
1	token suffix	*k* ending letters of each token are considered as a unit	Simplified Profile Intersection (SPI)	R7
1	token n-gram prefix	*k* starting letters of over-lapping token fragments are considered	Simplified Profile Intersection (SPI)	R8
1	token n-gram suffix	*k* ending letters of over-lapping token fragments are considered	Simplified Profile Intersection (SPI)	R9
2	BOW model	Reflects how the frequent usage of certain words by the user dominates other words	SPATIUM	R10
			L1 tf-idf	R11
2	Topic Modeling/ Folksonomy	Profile information related to the topics most frequently used by the user	NMF	R12
			tf-idf	R13
26	Number of occurrences of each letter (Aa-Zz)	Upper & lower case letters considered same	Cosine Similarity	R14
10	Number of occurrences of each numeric digit (0-9)	Digit Characters	Cosine Similarity	R15
1	Number of white spaces	White space is taken as one character	Cosine Similarity	R16
21	Number of occurrences of each special character	˜ @ # $ % ˆ & * - _ + = < > [ ] { }/\ |	Cosine Similarity	R17
8	Punctuation characters	, . ? ! : ; ’ "	Cosine Similarity	R18
8	Frequency of alphabetic, uppercase, lowercase, numeric, whitespace, special, punctuation & total characters	Presence of any one of them increments the counter	Cosine Similarity	R19
4	Number of occurrences of social networking characters	Count occurrences of @, #, RT, http/https (URL)	Cosine Similarity	R20
12	Probability of above characters being in first/ middle/ last part of text	4 characters (@ # RT http/https) and 3 positions (first/middle/last) - Average is taken	Cosine Similarity	R21
1	Total count of consecutive characters	Count of 3 or more characters occurring consecutively	Cosine Similarity	R22
13	Different character words (One, Two, ⋯, Twelve, More than twelve)	Count of words with different characters	Cosine Similarity	R23
1	Number of words	Count of total number of words in a tweet	–	
1	Hapax Legomena	Frequency of once-occurring words	Cosine Similarity	R24
1	Hapax dislegomena	Frequency of twice-occurring words	Cosine Similarity	R24
1	Number of occurrences of function words in each tweet	All function words taken together	Cosine Similarity	R25
1	Number of occurrences of dominant words in each tweet (excluding stopwords)	All dominant words excluding stopwords for a user taken together	Cosine Similarity	R26
1	All features (Combined)	All the features taken together	–	R27

On the contrary, handling the problem in an unsupervised manner involves learning the generalized patterns of all the compromised as well as genuine accounts and clustering them on-the-go based on the behavior being portrayed. Use of unsupervised learning is more efficient in authorship characterization and sometimes attribution tasks where the aim is to group together the samples belonging to similar characteristics. But for authorship verification problems, supervised learning i.e. classification is a more preferred method.

Though, recent works in the literature have demonstrated the use of machine learning based classification for authorship verification but still there is a discrepancy with the best set of classifiers applicable for the task. For example, Li et al. [12] evaluated that kNN is not suitable for AV but on the contrary, studies such as [14,15] emphasized on the effectiveness of kNN. Similarly, few works [10] have also encouraged the use of Support Vector Machines (SVMs) which are faster to train and scalable enough to large datasets for the task. Apart from that, type and count of data, choice of features, performance metrics have all been debatable in authorship verification tasks.

Hence, in this work most commonly deployed textual features have been supplemented with some topic modeling and folksonomic features for authorship verification and thereby identification of compromised accounts. Also, applicability of various machine learning classification techniques have been examined. An in-depth analysis has been made to determine the most appropriate classifier for the task.

4. Proposed work

This section discusses the proposed methodology adopted for the enhanced continuous authentication of social network messages. A supervised learning approach has been adopted which involves the common steps beginning from data collection, feature extraction and selection to classifier training and model evaluation.

4.1. Approach overview

This research aims to investigate and analyze the efficiency of textual features for authorship verification of social network messages. Authorship analysis problems are handled in two forms, using either a profile based approach or an instance based approach [39]. In the profile based approach, all the text samples of a user are concatenated to form a behavioral profile. Relevant features are extracted from this profile and the unknown text sample is then compared to the profile features to determine its authorship. On the other hand, in instance based approaches, an unknown text sample is compared to each text instance separately and answers are usually combined to give a final outcome.

Authorship verification technique used in this research work is a hybrid approach taking the concepts from both profile as well as instance based methods. The hybrid approach named TB-CoAuth followed in the work has been illustrated in Fig. 1. Tweets from a popular micro blogging site Twitter are used as text samples and the problem is studied as a two-class classification problem with tweets of same user acting as one class (positive) and spam/random tweets from other users as second (negative) class. For a new unknown tweet sample, the objective is to determine whether the activity under consideration has been performed by the concerned user X_i or not. The process can be divided into following steps.

1. **Interspersing:** For each user, a set of tweets is collected and placed in two sets namely, Baseline Set (S1') and Training Set (S2') in an interspersed manner. Interspersing has been preferred over direct partitioning in order to place related tweets concerning the similar subjects and trends in both the sets which help overcome the problem of seasonality. Baseline set (S1') represents the dedicated baseline profile of a user which is actually the concatenation of different tweet samples. All the further incoming tweets are matched against this behavioral profile to extract features and similarity measures. On the other hand, features extracted from training set (S2') help train the machine learning classifiers. Alongside these two sets, a negative set (NU') of the same size as S2' comprising of the spam and random tweets of other users is also collected.
2. **Textual Feature Extraction:** After pre-processing, relevant textual features as stated in Table 2 are extracted and profiled from all the tweet samples of a user. Each user must have enough samples so as to have sufficient amount of labeled training data.
3. **Numerical Feature Extraction:** For experiments, each tweet in the set S2 (Training Set) is compared against this baseline profile (S1) to compute the numerical features. As the tweets of the concerned user are split into the two sets, hence tweets in S2 belong to the concerned user and contribute towards building up of positive data samples. Collecting the ground truth data with negative samples consisting of the compromised tweets of users is a tedious task [40]. Textual behavior after compromisation is difficult to obtain as the tweets are either removed once a user gets to know that his account has been hacked or such tweets are not released in public. This constraint directed us to settle for the next best available course of action i.e. artificially compromising the accounts by manually injecting the randomness and spam posts. Existing works in this domain have also encouraged the use of the this practice [10,41,42]. Hence, an additional set NU that acted as negative class sample set is also created which comprise of the mixture of spam as well as random tweets from other users. Both positive and negative training data samples are compared against this profiled data to obtain similarity score values calculated using similarity metrics. These values further act as features which help train respective classifier.
4. **Feature Selection and Rank Aggregation:** Similarity features extracted are then fed to various feature selection algorithms to score and rank the features. Computed scores and ranks are utilized by the adopted rank aggregation procedure (BORDA) to generate a final ranking of features. Topmost *k%* features are selected for further computation.
5. **Supervised Machine Learning Classification:** Efficiency of various supervised machine learning classifiers is analyzed by performing the k-fold cross-validation on the collected data set. Parameters for each classifier are tuned using a train and validation set and the best fine-tuned classifier is deployed on test set to check the performance.
6. **Testing of new tweet samples:** Similar features extracted from the unknown sample are compared with both positive (User Ui) and negative (NotUserUi) learned patterns in the behavioral profile. Accordingly, with the help of suitable machine learning classifier, the unknown tweet sample is assigned the class having the most matched feature values. If the feature value does not match the positive class, it is counted as an anomalous behavior and the account is detected to be probably compromised. Unlike other domains, here the classifiers are trained and tested independently for each respective user because of the consistency maintained by a user on his/her own behavioral profile.
7. **Continuous Updating of Profiled data:** For incorporating the dynamics and changing behavior of users, an incremental learning process is adopted by continuously updating the training data with new patterns. Whenever a new upcoming tweet is predicted by the model to be from the concerned user, continuous updating in the baseline profile and training set will be performed in an interspersed manner. A FIFO approach needs to be followed in which once a new tweet is entered, the oldest tweet present in the original set is discarded in its place. Once, the efficiency of various models is checked on the train–test sets and an efficient model is selected, the updation policy is adopted which is then furthermore used throughout the deployment of the model. This updating in profile and usage of incremental learning is essential to be adopted to keep the process synchronized towards the user's writing style. Once people tend to change the subjects, and maybe even writing style over time, it is necessary to use the most recent posts from a user to recognize him. Thus, the model always discards the oldest posts used in exchange for the recently recognized ones.

4.2. Feature engineering

This work involves the use of both content-free and content-specific textual features for profiling and representing the writing style of a user. For content-specific analysis, n-gram, bag of words (BOW) and folksonomic representation have been studied whereas in content-free scenario stylometric features have been examined.

4.2.1. Feature extraction

For a tweet sample, both content-free as well as content-specific features are extracted and represented as a feature vector. Features have been normalized using RobustScaler that normalizes the features in the range of 0 to 1 by replacing the feature value using $val_{new} = (val - Q_1(val))/(Q_3(val) - Q_1(val))$ where Q defines the interquartile range. This normalization is preferred to reduce the effect of outliers present in the data. A total of 124 features have been extracted which are placed under 28 feature categories as shown in Table 2. Extracted features are broadly classified as content-specific and content-free features.

Content specific features

These features analyze the actual content of the message and examine meaning of text. Features explored in this category are n-grams (char and word n-grams), bag of words (BOW) and topic modeling features. In n-gram analysis both char as well as word n-grams have been extracted.

n-gram features process 'n' consecutive units of text together. Units could be anything ranging from characters and words to bits and bytes. In this paper, both character as well as word based n-grams have been studied. As evident from prior literature [14,15], char n-grams with n = 6 has shown good performance. Hence, in this work also, experiments have been performed with n = 6. On the other hand, for word n-grams, both unigrams (n = 1) as well as bigrams (n = 2) have been examined.

token prefix and suffix are another set of features considered in this research work. Tokens here represent the segmented character based units where token k-prefix state the *k* starting letters of each token. For example, the word *compromise* may be tokenised as: →{c, co, com, comp, compr, compro, ... } depending on the value of *k*. Similarly, token k-suffix define the *k* ending letters of tokens. For example, depending upon the value of *k*, the word compromise can be tokenised as →{e, se, ise, mise, omise, ... }.

Considering a sentence '*This account has been compromised*', its respective token 3-prefix and 3-suffix would be:
token 3-prefix: { Thi acc has bee com }
token 3-suffix: { his unt has een sed }

In this research work, textual features respective to the token k-prefix and k-suffix have been extracted taking $k = 3$.

token n-gram k-prefix and n-gram k-suffix also account for the usage of *k* starting and ending letters of overlapping token fragments respectively. For example, with n = 3 and k = 2, for the sentence, '*This account has been compromised*' respective token n-gram k-prefix and suffix is as follows:
token 3-gram 2-prefix: { Th_ac_ha, ac_ha_be, ha_be_co }
token 3-gram 2-suffix: { is_nt_as, nt_as_en, as_en_ed }

Similar to the above stated examples, for experiments in this research work, $n = 3$ and $k = 2$ have been taken.

With **Bag of words** feature model, a data-driven method is used in which frequency count of the most frequently occurring words is analyzed using two type of frequency features namely, tf-idf (term-frequency inverse document frequency) and SPATIUM-L1 [43]. tf-idf measures the count of occurrence of a word in a tweet alongside the count of tweets in which the word occurs. Similarly, a distance measure called SPATIUM-L1 as defined in Eq. (1) computes the distance value using occurrence probability of the frequent term in the given tweets.

$$SL(U,V) = \sum_{j=1}^{h} \left(P_U[w_j] - P_V[w_j]\right) \tag{1}$$

where 'U' and 'V' denote the tweet in question and baseline set respectively. Also, w_j defines each frequently occurring term amongst a total of 'h' frequent terms with P_U and P_V representing the probabilities.

Topic Modeling features are used to profile the information related to the popular topics tweeted by a user. This has been performed using folksonomic (hashtags) and Non-Negative Matrix Factorization (NMF) concepts. Folksonomy also referred as social tagging is used to describe the content of a web related document (a blog, picture, video, tweet) by annotating it with the user defined tags. Two popular Twitter entities namely hashtags and mentions reflect the folksonomy concept. Frequency counts of popular hashtags/mentions and NMF extracted topics are profiled and compared against the incoming tweets using similar tf–idf concepts as in Bag of words model. Hashtags have been fetched from the tweets using a python script. NMF algorithm defined under matrix decomposition module of Python machine learning repository, scikit-learn, has been used to extract topics from the tweets. Extracted topics are referred as tags in folksonomic terms. The input to the algorithm consists of a document term matrix (DTM) where terms are extracted from the baseline profile S1 and the documents are the tweets (textual features) in set S2. This helps to analyze both terms and the documents i.e. terms and the tweets in which the term appear. The output is a list of topics, each represented as a list of terms.

Content free features

Second set of text based features explored in this work are **stylometric features**. Content specific features focus on the meaning of the text whereas content free features on the other hand deal with how the content is written i.e. analyze the writing style patterns of users. Both lexical and syntactic stylometric features are used for the task. Lexical features analyze the frequency of occurrence of various words, characters, punctuations, white spaces, legomenas and other vocabulary richness measures whereas syntactic features include the punctuations, function words, parts of speech etc.

Similarity/Comparison Measures

In this work, machine learning classifiers are fed with the numerical attributes which are obtained by applying the appropriate similarity measures on the textual features. Table 2 presents the feature and the corresponding similarity metric used. Similarity is computed taking each tweet in training and testing set as data tweet and all the tweets in baseline set as reference set.

Simplified Profile Intersection (SPI): SPI is a commonly deployed intersection measure used to define the count of common units between any compared sets and is calculated as follows:

$$SPI(U,V) = |N(U) \cap N(V)| \tag{2}$$

where U and V are two sets with some textual elements. SPI can attain a minimum value of 0 indicating that there is nothing common in between the sets whereas the maximum value equivalent to the size of the smallest set out of U and V indicating that the bigger set contain whole of the smaller set.

Jaccard coefficient (JC): It is also a similarity measure defined over the intersection (common) and union (all) of elements present in the sets. Again for two textual sets U and V, Jaccard coefficient is calculated as:

$$JC(U,V) = |N(U) \cap N(V)|/|N(U) \cup N(V)| \tag{3}$$

As Jaccard coefficient is directly proportional to the count of common elements, higher the shared elements, higher is the Jaccard coefficient. A user typically adhere to his/her profile, hence, for the authorship verification task, a text from the same user is expected to have higher JC than from other users.

tf-idf: *Term frequency-inverse document frequency (tf-idf)* [44] measures the count of occurrence of a word in a tweet alongside the number of tweets in which the word occurs. It is a significant measure especially in authorship verification scenario as it keeps into account both the tweet in question and the other extracted tweets of the user. tf-idf helps to evaluate the importance of each word by representing text as a vector space model. With vector representation of baseline profile, for each term, the term frequency, tf, (frequency of occurrence of a term v in the given tweet t) and inverse document frequency, idf, (logarithm of inverse of number of tweets the term v occurs at least once) is used to compute tf-idf as follows:

$$tf\text{-}idf(v,t) = tf(v,t) * idf(v) \tag{4}$$

where, $tf(v,t) = \frac{count\ of\ \mathbf{v}\ in\ tweet\ t}{Number\ of\ words\ in\ tweet\ t}$ and $idf(v) = log(\frac{N}{df+1})$. Here, N denote the total number of tweets (i.e. number of tweets in Set S1') and df denote the number of tweets in Set S1' containing the term v.

After sorting the computed tf–idf scores in ascending order and ranking accordingly, the top ranked terms are picked and stored as frequently occurring terms in the users profile. These terms are then used against test sets (S3' and SR') to compute tf–idf values further used as features with target labels 'SameUser' and 'NotUser' respectively.

Cosine Similarity: As stylometric features deal with frequency counts, hence, cosine similarity method is used to compute similarity by considering frequency features as vectors. Computation of cosine similarity also take the orientation of vectors into account. In order to compute cosine similarity, the cosine between the vectors is computed using the dot product as follows:

$$A \cdot B = \|A\|\ \|B\| cos\theta \tag{5}$$

Further, using this dot product, the cosine similarity is computed as defined in Eq. (6)

$$Similarity = cos\theta = \frac{A \cdot B}{\|A\|\ \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{6}$$

Higher the cosine similarity, closer are the documents (tweets) by angle (orientation).

SPATIUM-L1: Unlike other discussed measures, SPATIUM-L1 is a distance measure that computes the distance value using occurrence probability of the frequent term in the given tweets. SPATIUM-L1 is computed using the formula mentioned in Eq. (1).

As it is a distance measure, hence a lower value is expected and preferred for the same user and higher for the other users.

4.2.2. Feature selection

Presence of redundant and irrelevant features for classification tasks may result in the reduced performance, hence, it is a good practice to select only the most relevant features. Alongside, the removal of irrelevant and redundant features may also reduce extra processing overhead for training and classification. A feature selection technique mark and rank each feature according to its correlation to other features or class label. But the selection of appropriate feature selection technique is itself a topic of concern. Instead of dealing with the best features obtained by each feature selection technique, rank aggregation using BORDA [45,46] has been performed for aggregating the ranks obtained by the following feature selection techniques.

t-score: It is a well suited statistical method for feature selection especially designed for binary classification problems. t-score help select features that make the mean of both binary classes different. For each feature f_i, suppose μ_1 and μ_2 represent the mean feature values for the instances from two classes, σ_1^2 and σ_2^2 denote the corresponding variances, and n_1 and n_2 defines the number of instances from these two classes. Then, ratio of mean and variance difference of test samples in each class is used to calculate t-score for the feature f_i as follows:

$$t - score(f_i) = \frac{|\mu_1 - \mu_2|}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}} \tag{7}$$

t-score basically helps to assess, whether for a concerned feature its mean values for observations belonging to separate classes are statistically different. Higher the mean difference, more distinguishable are the classes, hence, better is the feature.

Gini-Index: It is a correlational statistical feature selection method designed to choose features that distinguish different classes. Gini Index uses the concept of impurity (considering all the possible occurrences of a feature) which is calculated as:

$$Gini_{score}(X) = 1 - \sum_{t=1}^{2} P_t^2 \tag{8}$$

where P_t defines the probability that a tuple in the set X belongs to class C_t and is calculated using the conditional probability $|C_t, X|/|X|$. As stated, this sum is calculated over all the classes (as authorship verification is studied as a binary classification problem, only two classes are considered). All the available values of a feature are used to compute the $Gini_{score}$ of a feature which is further subtracted from the calculated impurity. For a feature f_i, with v different values, let X and $\overline{X}$ represent the instances with values of feature less than or equal to some jth feature value and the ones with values larger than the jth feature value respectively. This help in the separation of the dataset into two parts X and $\overline{X}$ using the jth feature value. Gini index score for the concerned feature f_i is further calculated as:

$$Gini(f_i) = \min_X \left(p(X)(Gini_{score})(X) + p(\overline{X})(Gini_{score})(\overline{X}) \right) \tag{9}$$

The above equation can be further expanded as follows:

$$Gini(f_i) = \min_X \left(p(X)(1 - \sum_{t=1}^{2} p(C_t|X)^2) + p(\overline{X})(1 - \sum_{t=1}^{2} p(C_t|\overline{X})^2) \right) \tag{10}$$

where, $p(.)$ represents the probability. For example, $p(C_t|X)$ denotes the conditional probability of class t given the instances in X. Among the computed values for each i, the minimum value is chosen as the Gini Index for that feature. Keeping impurity into consideration, lower values of Gini Index are preferred.

Fisher score: It is a similarity based feature selection method selecting features with high intra-class similarity but less inter-class similarity using the following Fisher score formula:

$$Fr_{score}(f_j) = \frac{\sum_{t=1}^{2} n_t(\overline{x_{jt}} - \overline{x_j})^2}{\sum_{t=1}^{2} n_t \sigma_{jt}^2} \tag{11}$$

Again being a binary classification problem, number of classes (t) is fixed to 2 i.e. value of 't' is varied as 1 and 2. n_t defines the count of samples in class t and $\overline{x_j}$ and $\overline{x_{jt}}$ indicate the mean value of feature f_j and that of samples in class t respectively. σ_{jt}^2 represents the variance of a feature f_j for all the sample values in class 't'. Features are ranked according to their Fr_{score} and features with higher values of Fr_{score} are considered better.

Correlation Feature Selection (CFS): This feature selection method select the subset of features which have strong feature-class correlation ($corr_{cf}$) and weak feature-feature correlation ($corr_{ff}$)

i.e. features having good correlation with the class but not with other features. $\overline{corr_{cf}}$ and $\overline{corr_{ff}}$ represent the average value of all feature-class correlations and feature-feature correlations respectively. For a feature subset (S), CFS_{score} defining the influence of a particular subset is computed as follows:

$$CFS_{score}(S) = \frac{h\overline{corr_{cf}}}{\sqrt{h + h(h-1)\overline{corr_{ff}}}} \tag{12}$$

where h denotes the number of features selected in that subset S. In case of correlation feature selection, higher CFS score indicates a better feature. A better way to represent the correlation feature selection is to define it this way:

$$CFS = \max_{S_h}\left[\frac{corr_{cf_1} + corr_{cf_2} + \cdots + corr_{cf_h}}{\sqrt{h + 2(corr_{f_1f_2} + \cdots + corr_{f_if_j} + \cdots + corr_{f_hf_{h-1}})}}\right] \tag{13}$$

The correlations $corr_{cf}$ and $corr_{ff}$ are computed using the symmetrical uncertainty (su) function (present in skfeature library in python) that make use of the Information Gain (IG) and Entropy (H) as follows:

$$su(f1, f2) = 2 * IG(f1, f2)/(H(f1) + H(f2)) \tag{14}$$

Finally, the subset of features with maximum Correlation Feature Selection Score is chosen.

4.2.3. Feature fusion/ rank aggregation

Rank Aggregation also termed as feature ensemble approach involves the fusion of ranks generated by different feature selection techniques [47,48]. Technically, Rank Aggregation can be defined as

> *"For a given set of objects $o_1, o_2, \ldots, o_n$ and their corresponding rankings $r_1, r_2, \ldots, r_n$, a single ranking **r** is to be produced that is in agreement with the existing rankings"*

In this work, objects correspond to the different feature selection algorithms and the rankings to the respective ranks given to the features by each method. As each feature selection technique incorporates different criteria for selecting features, hence aggregating ranks produced by different techniques enables us to analyze different aspects of data considering numerous views of the importance of features. Moreover, ensemble of rankings from multiple techniques help to study the complementaries of various methods and improve the selection by limiting the influence of every single technique.

Literature reveals the existence of numerous methods to combine the rankings produced by different techniques such as aggregation [49], majority voting [50,50], Borda [45,46], Schulze [51] and many others [52,53]. In the literature, no consensus is found about which method to prefer over the other. This study involves the use of a popular rank aggregation method named BORDA [45, 46] which has been commonly deployed for the rank aggregation task in many inter disciplinary domains.

BORDA method: BORDA [45,46] utilizes a positional voting system by assigning points to each feature according to its ranking position. Points are placed in an increasing order with first ranked feature given lower points and the feature ranked last attaining the maximum points. During decision process, the feature having lowest aggregated points is ranked and hence considered better. The procedure adopted for the same has been illustrated in Algorithm 1.

Rather than considering the best feature from this rank aggregation technique as a standalone feature, experiments have been performed by varying the selected features.

Algorithm 1 BORDA Algorithm for Rank Aggregation

```
Input: Rankings produced by respective feature selection algorithm (r1, r2, ... $r_m$)
Output: Ranks array corresponding to the ranking produced by BORDA aggregation
    Initialization :
    /* n is the number of features to be ranked /*
1: for $i = 1$ to $n$ do
2:     feature[i] = 0
3: end for
    Assignment of positional weights to each feature
4: for $i = i$ to $n$ do
5:     feature[i] = $pos(r_1(i)) + pos(r_2(i)) + pos(r_3(i)) + \ldots + pos(r_m(i))$
6: end for
    Sort the feature array in ascending order of positional weights
7: Ranks = sort(feature)
8: return Ranks
    /* $pos(r_j(i))$ defines the position of ith feature in $r_j$ ranking */
```

4.2.4. Classification approach

From the feature vector obtained as a result of rank aggregation, top 'k' features are used to train the corresponding classifier. From the related works deploying machine learning classification approaches for the task, the classifiers namely, k-Nearest Neighbor (kNN), Random Forest (RF), Gradient Boosted (GB), Support Vector Machine (SVM) and Multilayer Perceptron (MLP) are found to perform well for text classification as well as authorship verification problems [10,12–15,54,55]. Hence, efficiency of the stated classifiers have been analyzed in this work. Table 3 gives a brief description of the said classifiers as well as the effect of parameter variation on performance.

k-NearestNeighbor (kNN): kNN is an incremental learning classifier which unlike other classifiers do not learn patterns in the data but rather determines the target label for the data at run time by calculating distance of the data point with other 'k' number of training samples in the data. Label of the unknown data is predicted as the target of the majority of nearest neighbors. Though choice of the parameter 'k' (neighbors) is data dependent but in general, lower values of 'k' overfits and makes the model complex whereas higher values makes the model too generalized. Both the cases lead to decreased efficiency and hence, an optimal choice of 'k' is always desired.

Random Forest (RF): Random forest represents an ensemble of decision trees where randomness is inserted in each decision tree to firstly, make each tree slightly different and secondly, avoid overfitting usually inserted by a single decision tree. It is also referred as meta estimator that uses average of results obtained by individual decision trees over different sub-samples to improve the performance. Main parameters of consideration for Random Forests are *n_estimators* (number of trees), *max_features* (number of features for each split), *max_depth* (maximum depth of each tree). Effect of varying these parameters on performance is described in Table 3.

Gradient Boosted (GB): Gradient boosted/boosting is again a decision tree ensemble method used to optimize the performance of individual decision trees. It involves the combination of various independent shallow decision trees to create an optimized model incorporating improved performance as compared to a single decision tree. Unlike random forests, no randomization is inserted in gradient boosted and trees are build in a serial manner where mistakes of one tree are corrected by the next one. Depth of each tree is usually kept low in order to conserve memory but perform fast prediction. Gradient Boosted is highly dependent on parameter tuning but among various parameters the main influential ones include *learning_rate* (how strongly a tree could correct mistakes of previous trees), *max_depth, n_estimators*.

Support Vector Machines (SVM): SVM, a discriminative classifier works by finding an optimal hyperplane to distinguish data points of different classes and minimize the error. One of the important task in SVM is to select a maximum marginal hyperplane i.e. a hyperplane that maximizes the margin between the support vectors (closest points to the hyperplane). Secondly, varying kernels help to make a choice for both linear as well as non-linear decisions by transforming and expanding the input space to a higher dimensional space. Three crucial parameters mainly optimized for SVM are *C, gamma and kernel*. *C* is a regularization parameter used to avoid overfitting and limits the importance of each feature. *gamma* on the other hand is defined for non-linear kernels and states the width of kernel radius which help control complexity of the model and *kernel* helps incorporate non-linear decision boundaries.

Multi Layer Perceptron (MLP): It is a category of feed-forward neural networks and uses multiple stages of experiments at back end for decision making. It is an advancement of linear models such as logistic regression where input and output layers are complimented with some non-linear hidden layers in between. Major building block of any neural network is the neurons and neurons in the input layer represent the input features. Each hidden layer in the network also consists of neurons which performs the weighted summation of the values from the previous layer followed by transformation using some *activation function*. Therefore, parameters of paramount importance in MLP are *number of hidden layers, hidden_layer_sizes, activation_function* and *learning_rate*. Impact of varying these parameters on the model is discussed in Table 3.

5. Experimental results and discussion

This section discusses the experiments performed on the extracted features, application of feature selection, rank aggregation and then the corresponding classification approach to evaluate the efficiency. All the performed experiments are coded in Python. Python's machine learning toolkit called scikit-learn [56] has been used for the classification tasks. It is pertinent to mention that the proposed approach is applicable only on the active users i.e. users that frequently use the social network platform.

5.1. Data collection

Basic prerequisite before performing any experiments is the collection/creation of data set. Among various prevalent social networks, Twitter has always been the gold standard chosen by researchers for experiments because of the accessibility and flexibility of API services. In the literature, time ordered data of a large set of Twitter users (approximately, 147K users) has been collected by Li et al. [57] but it is limited to the availability of only 500 tweets of each user. Furthermore, the collected tweets are from the year 2012.

Though approach designed in this work is a generalized one and is independent of the time and source of data, still in the subsequent years, Twitter as a platform has undergone various modifications such as increase of tweet size from 140 to 280 characters, introduction of media tags, image and video links not counted as character limit, enabling retweet on one's own tweet etc. In the said dataset, the number of tweets collected from each user are less and insufficient which may limit the amount of data for profiling and train–test sampling, hence, it was preferred to conduct experiments with a comparatively large amount of latest tweets to strengthen the stated claims. Usernames of a set of 4000 users are randomly chosen from the available 147K users and 4000 latest tweets for each user (starting from 23 February, 2019 till the 4000th historical tweet of that user) were extracted using the Twitter API. Users with less than 4000 tweets are discarded which left us with 3057 users for experiments. The count of users and tweets is made taking into consideration the availability of resources for conducting experiments. But this nowhere limits the approach on these parameters as each user is independently analyzed hence, experiments can be performed with any number of users.

5.2. Data preprocessing

Because of the limitation of 140/280 characters of tweets, two tweets are combined to form a slightly larger text for easy and fair analysis. Also, tweets involving UNICODE blocks are removed. Other preprocessing steps involve conversion of each tweet in lower case followed by removal of punctuation and stop-words. For stylometric features no preprocessing is performed as presence of every minor detail has an influential impact on performance.

5.3. Experimental approach and results

Varying the hyper parameters of classifiers, 90 experiments are performed. Experiments are performed on a 64-bit Windows Operating system with 4.2 GHz i7 processor and 32 GB RAM. The collected data is placed in an interspersed manner into two sets i.e. 50% data in Baseline set and 50% in training set. As a common machine learning classification policy, the training set data is split into 60% training and 40% testing parts. A Shuffle-Split approach with 5-fold cross-validation is performed on training data (split into 60–40 train and validation set). Every classifier is parameter-tuned and trained with the feature vector obtained from the top k features from BORDA technique. Top Ranked features selected using each independent feature selection technique as well as BORDA rank aggregation is shown in Fig. 2.

It describes the frequency count in terms of number of users for the top ranked feature i.e. number of users with respective feature as the top ranked feature. As each feature selection technique has a different policy of selecting features, hence, it lead to the selection of different features. To overcome the ambiguity, aggregation of features is performed using BORDA and the ranks generated using BORDA are considered for further experiments. After generating the ranks, it is usually practiced to select top 'k' features and perform experiments using them. Keeping computational time as well as resource constraints into consideration, to decide for the value of 'k', feature importance experiments have been performed on the data of randomly chosen 500 users by varying the value of 'k' and choosing top k features every time for the experiments. As shown in Fig. 3, it has been observed that for each classifier, with the increase in number of features, performance improved but there comes a point ($k = 9$ or 10) where adding more features deteriorates the performance. Analysis till 14 features has been done as performance after 9 or 10 features was seen continuously decreasing. Hence, the value of k beyond which performance of classifiers deteriorates has been seen to be top 9 features for GB (*Accuracy: 96.58%, Precision: 96.93%, Recall: 96.17%, Fscore: 96.54%*), SVM (*Accuracy: 95.60%, Precision: 96.54%, Recall: 94.61%, Fscore: 95.53%*) and MLP (*Accuracy: 90.90%, Precision: 90.32%, Recall: 92.05%, Fscore: 90.93%*) whereas top 10 features for kNN (*Accuracy: 95.63%, Precision: 96.73%, Recall: 94.37%, Fscore: 95.59%*) and RF (*Accuracy: 96.14%, Precision: 97.00%, Recall: 95.15%, Fscore: 96.06%*). Performance degradation for values

Table 3
Impact of parameter settings on different classifiers.

Classifier	Highly varied parameters	Behavior of parameters	Remarks/Key points
K-Nearest Neighbor	• *n_neighbors*	• ↑ neighbors – smoother boundary – ↓ model complexity (simple model) • ↓ neighbors – ↑ model complexity – ↓ generalized • Preferably odd values are taken	• decide majority class from the 'k' neighbors
Random Forest	• *n_estimators* (number of trees) • *max_features* • *max-depth*	• ↑ max_features – similar trees — every tree utilize similar important features • ↓ max_features – different trees – deep analysis of selected features for decision making — reduces overfitting • ↑ n_estimators – better (but adjust as per availability of time and memory)	• Efficient performance on large datasets • No need of data scaling • Perform quite well with default parameters (does not require heavy parameter tuning) • Noteworthy performance on large datasets • Not efficient for High-dimensional , sparse data
Gradient Boosted	• *random_state* (Seed to generate same random numbers) • *n_estimators* (No. of trees to model) • *max_depth* (Max depth of a tree) • *learning_rate* (decide impact of each tree on final decision (controlling magnitude change in estimates)) • *min_samples_split* (Minimum nodes required for splitting) • *min_samples_leaf* (Minimum samples required in a leaf node) • *max_features* (Number of features for best split)	• Important to set random_state to avoid generate different trees • Model is robust to value of n_estimators • ↓ learning rate — require more trees – ↑ computation cost • ↑ learning rate or ↑ n_estimators – stronger connections – complex model – leads to overfitting • ↑ max_depth or ↑ max_features — overfitting • ↑ min_samples_split — underfit model	• Independent of data scaling • Automatic Feature Selection • High training time • Not suitable for high-dimensional sparse data • Suitable for datasets having binary as well as continuous variables • Parameter tuning is important.
Linear SVM	*C* (Regularization parameter)	↓ C – restricted model	• Very sensitive to the scaling of data (Normalized data) • Require tuning of parameters for improved performance • Not efficient on low-dimension data (few features)
Kernalized SVM	• *C* • *Kernel* • *gamma*	• ↓ gamma – less variation in decision boundary – simple model • ↑ gamma or C – complex model – overfitting	• Reflect good performance on both low and high dimensional data • Scaling issues with large datasets • Suitable for homogeneous data
Multi Layer Preceptron (Neural Networks)	• *activation function* • *alpha* (regularization parameter) • *number of hidden layers* • *hidden_layer_sizes*	• ↑ hidden units – smoother boundary • ↓ alpha – ↓ regularization	• Initial weights are chosen randomly, hence may impact results for small datasets • Scaling of data is must (data to have zero mean and 1 variance) • low weight features for all the hidden units signifies them to be less important • Suitable for homogeneous data

Symbols: ↑ **increase**, ↓ **decrease**.

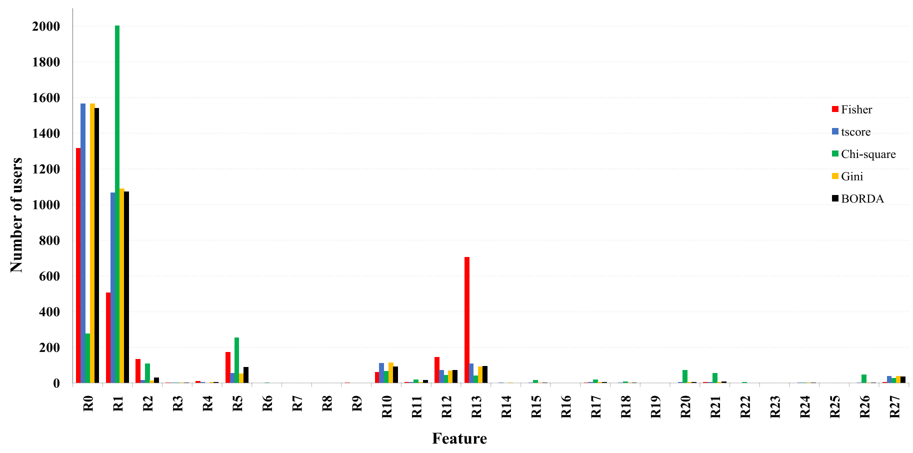
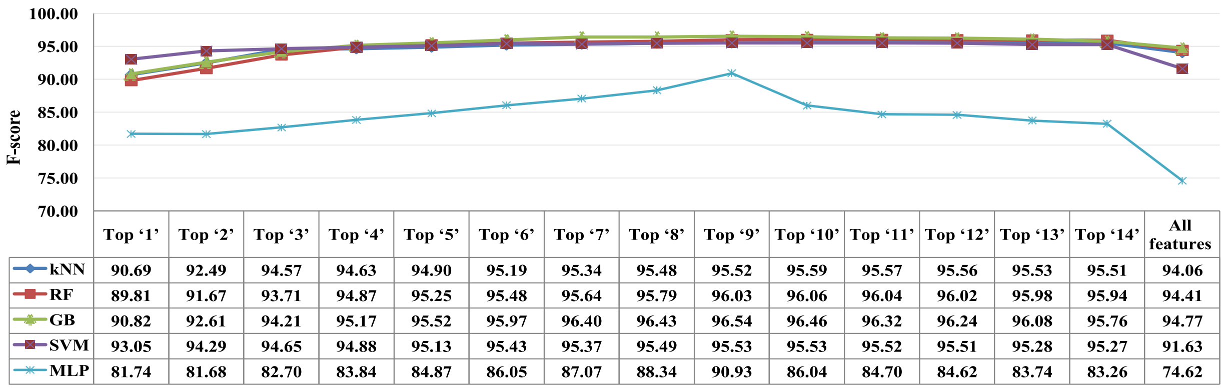


Fig. 2. Top Ranked features using different feature selection techniques and BORDA in terms of count of users.

of k to be 9 and 10 in case of kNN and RF is almost negligible and hence for further computation, k = 9 has been taken for all the classifiers i.e. experiments have been performed with top 9 features. For a deeper analysis, proportion of each feature amongst the top 20% features can be observed from Fig. 4. Fig. 4 also signifies the number of users having a feature at a particular rank position. As an example, from 3057 users, 3004 users have feature R0 in their top-5 features with 1541 users ranking it as 1st and 31 users ranking it as 5th. Availability of R0 beyond rank 5 is almost negligible. Similar is the case with other features.



	Top ‘1’	Top ‘2’	Top ‘3’	Top ‘4’	Top ‘5’	Top ‘6’	Top ‘7’	Top ‘8’	Top ‘9’	Top ‘10’	Top ‘11’	Top ‘12’	Top ‘13’	Top ‘14’	All features
kNN	90.69	92.49	94.57	94.63	94.90	95.19	95.34	95.48	95.52	95.59	95.57	95.56	95.53	95.51	94.06
RF	89.81	91.67	93.71	94.87	95.25	95.48	95.64	95.79	96.03	96.06	96.04	96.02	95.98	95.94	94.41
GB	90.82	92.61	94.21	95.17	95.52	95.97	96.40	96.43	96.54	96.46	96.32	96.24	96.08	95.76	94.77
SVM	93.05	94.29	94.65	94.88	95.13	95.43	95.37	95.49	95.53	95.53	95.52	95.51	95.28	95.27	91.63
MLP	81.74	81.68	82.70	83.84	84.87	86.05	87.07	88.34	90.93	86.04	84.70	84.62	83.74	83.26	74.62

Fig. 3. Performance analysis of different classifiers by taking top ‘k’ features from BORDA.

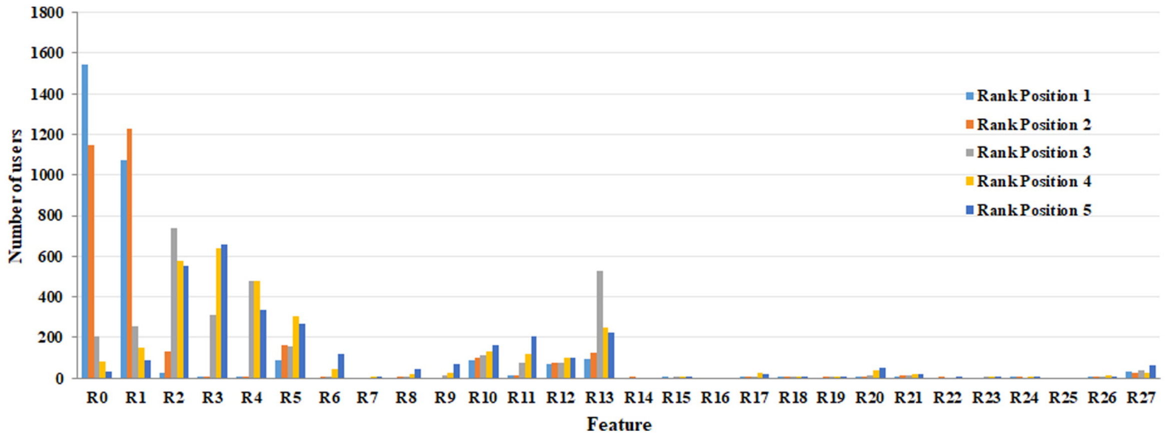


Fig. 4. Number of users with the respective feature in top 5 positions.

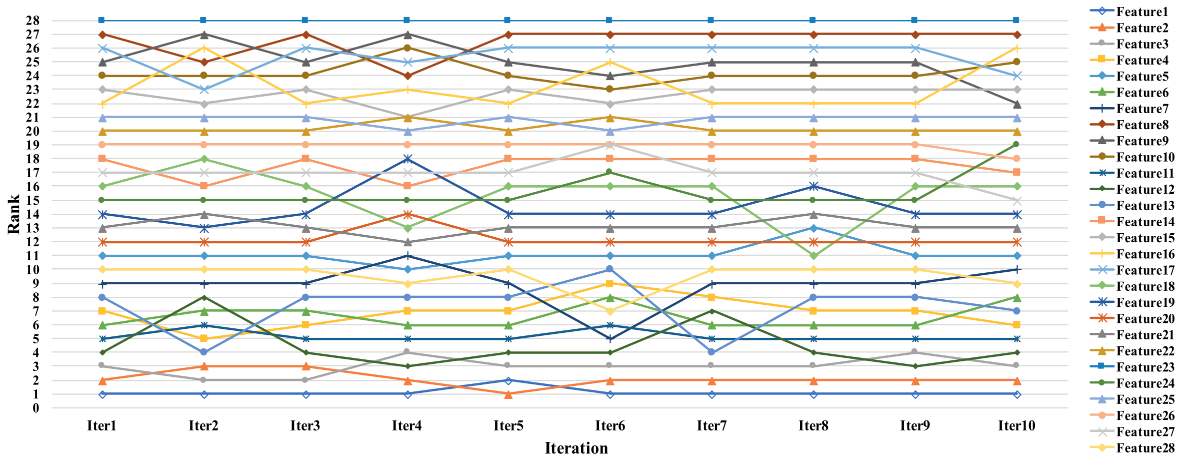


Fig. 5. Rank position of each feature in the random 10 iterations of a user.

While handling the problem from binary classification perspective, the requirement of a balanced two-class data is necessary. Tweets of the same user act as the positive class whereas random collection of spam tweets or that of the other users act as the negative class. It is important to mention here that though, every user considered for experiments has different set of negative samples, but the random placement of tweets in each user’s sample allows us to use any such sample. Still in order to avoid the ambiguity, we considered taking various random negative samples for all users and performed 100 iterations of experiments for each user. All the methodological steps were performed for each sample. In other words, for each iteration, initially, feature selection steps were performed followed by the performance evaluation. In case of feature selection, it was observed that almost similar ranking of features was produced using BORDA for each user in every iteration. Even in cases of deviations in feature rankings, it was observed that same set of features were there in the top ‘k’ features with a difference of few ranks. As experiments using the prior training set were conducted using top ‘k’ features hence overall for each user there was hardly any variation in the top ‘k’ features obtained for every iteration. This can be observed from Fig. 5 that depicts the rank position of each feature in the random 10 iterations for a random user.

Table 4
Average Performance of different classifiers for all users over 100 different training datasets.

Classifier	Accuracy (Avg)(%)	Precision (Avg)(%)	Recall (Avg)(%)	F-score (Avg)(%)	ROC_AUC (Avg)(%)
kNN (9)	92.07 ± 3.9	93.17 ± 3.8	90.83 ± 6.1	92.42 ± 4.7	93.85 ± 4.2
RF (30,'auto',100)	93.33 ± 3.8	93.49 ± 4.2	92.19 ± 5.0	92.82 ± 4.4	94.20 ± 3.9
GB (0.1,1,200)	93.46 ± 3.7	93.84 ± 3.9	92.03 ± 5.3	92.89 ± 4.3	94.24 ± 4.0
SVM (10,1,'rbf')	94.22 ± 3.6	94.28 ± 3.7	93.01 ± 5.8	93.04 ± 4.5	95.27 ± 3.9
MLP('tanh',0.05, (50,100,50),'adaptive')	92.36 ± 4.2	92.03 ± 4.7	90.55 ± 6.5	91.68 ± 5.1	94.14 ± 4.5

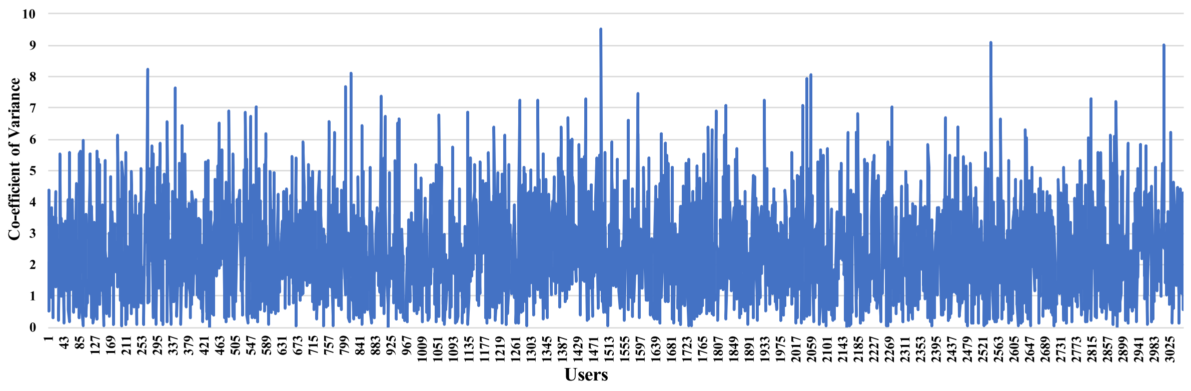


Fig. 6. Deviation in F-score performance by SVM over 100 iterations of varied test samples.

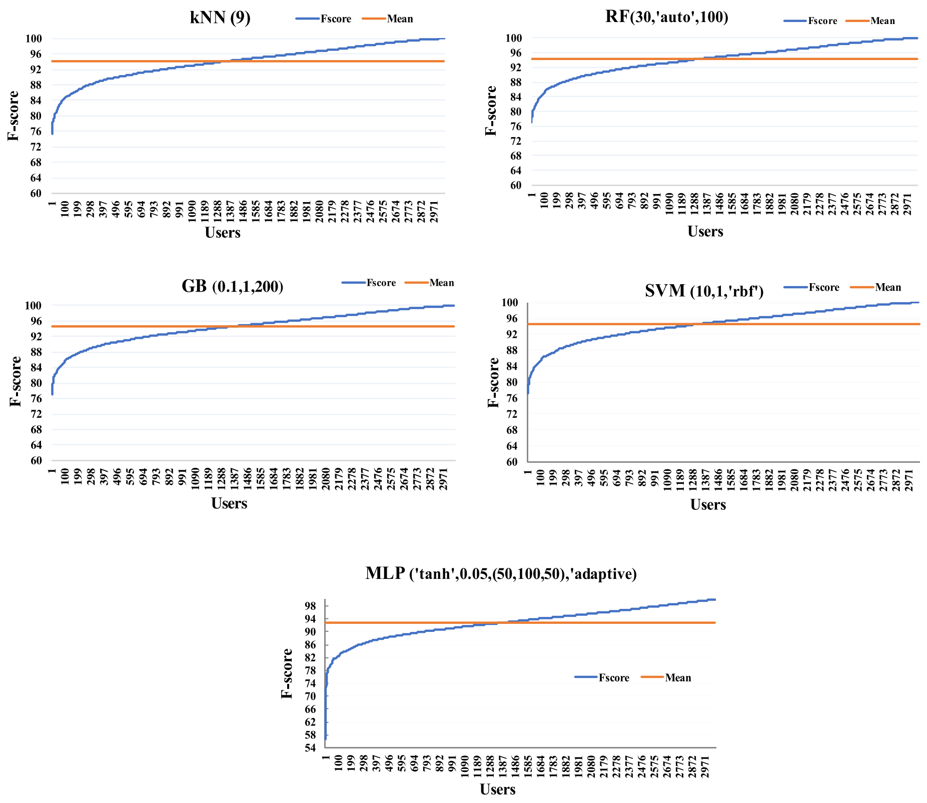


Fig. 7. F-score performance of 3057 users on the train data.

Firstly, in terms of performance evaluation, almost similar performance was obtained for each user using other negative samples also. The probable reason for the same could be the difference in textual patterns of positive and negative class data that might have helped the machine learning models to easily learn the patterns based on the optimal features in place. Secondly, the usage of feature selection as well as rank aggregation techniques for the optimal selection of features, help in building ideal models. Third probable factor for the same is the tuning of hyper-parameters for each respective machine learning model that really helped in selecting highly efficient models for experiments. Table 4 presents the quantitative analysis of the average score of performance metrics over 100 iterations for all users i.e. the validation results are averaged over the different iteration rounds. Alongside average values, co-efficient of variance (CV) is also reported that signifies the relative variability in values.

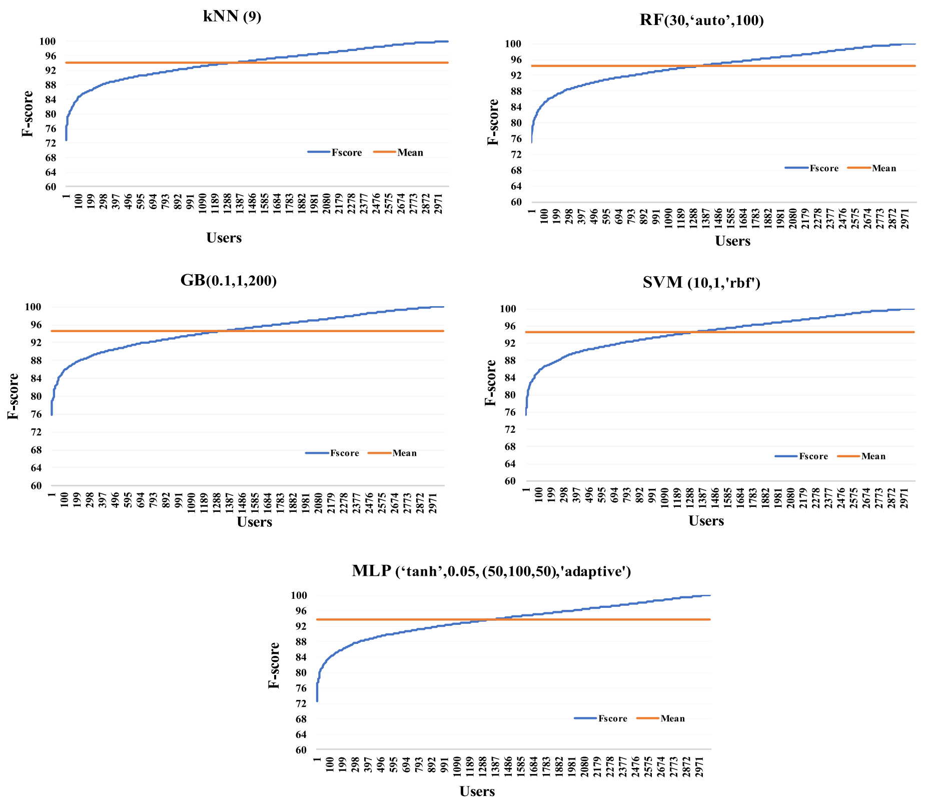


Fig. 8. F-score performance of 3057 users on the test data.

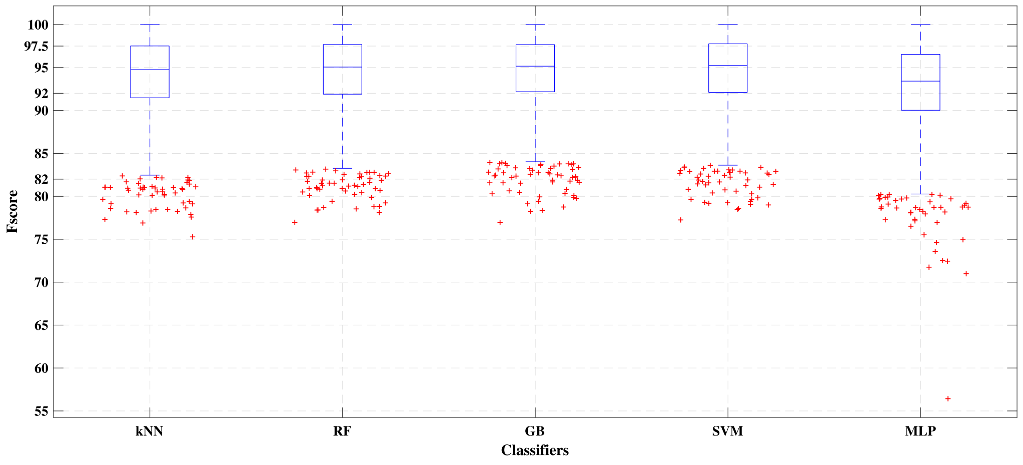


Fig. 9. Fscore variation with different classifiers considering all the 3057 users.

It is computed using the standard deviation and mean values of the respective performance metric. With the top *k* features, once a classifier is trained, test data is fed to each parameter-tuned classifier to validate the model and obtain the evaluation metrics. Again to avoid the abstruseness, trained classification models for each user have been tested on the randomly chosen 100 test samples. Fig. 6 represents the deviation in performance (in terms of Co-efficient of variance CV of F-score) for each user over the 100 iterations. It is evident from Fig. 6 that for most of the experimented users, F-score deviations varied below 5%–6% in all the 100 iterations. Even the users who had deviations above 5%–6% are mostly the ones who does not adhere to their textual profiles otherwise as well. Overall performance of the discussed classifiers have been analyzed on the basis of various evaluation metrics and is shown in Table 5. Performance metrics obtained for both train and test set have been presented.

Reported average scores correspond to the mean of the performance obtained by all the individual users. In case of train set performance, average score is calculated by first computing the average of the 5-fold cross-validations obtained for each user on the respective training set and then taking the mean of reported scores for all the users. For test set, average score represents the average of the test set performance of all individual users on the respective test set.

Performance obtained by each user on its respective training and test set data has also been presented in Fig 7 and 8 where F-score respective to each user has been plotted. F-scores obtained by different users have been sorted and then plotted to avoid a messy graph.

Table 5
Performance of different classifiers with their optimized parameters.

Classifier	Optimized parameters	Best parameter setting	Fit time (s)	Accuracy (Avg)(%)		Precision (Avg)(%)		Recall (Avg)(%)		F-score (Avg)(%)		ROC_AUC (Avg)(%)	
				Val	**Test**	Val	**Test**	Val	**Test**	Val	**Test**	Val	**Test**
kNN(neighbors)	**Neighbors:** [3,5,7,9]	**kNN**(9)	0.28	94.21 ± 4.2	92.29 ± 4.3	95.55 ± 3.7	92.32 ± 4.0	92.68 ± 5.1	91.84 ± 5.3	94.08 ± 4.3	92.15 ± 4.5	97.49 ± 2.3	94.29 ± 4.3
RF(max_depth, max_features, n_estimators)	**max_depth:** [10,20,30] **max_features:** [‘auto’] **n_estimators:** [10,100]	**RF**(30,‘auto’,100)	1.02	94.39 ± 4.1	92.45 ± 4.2	94.88 ± 4.0	92.88 ± 4.3	93.85 ± 4.4	91.94 ± 4.6	94.35 ± 4.2	92.39 ± 4.3	98.01 ± 2.1	94.45 ± 4.2
GB(learning_rate, max_depth, n_estimators)	**learning_rate:** [0.01,0.1] **max_depth:** [1,2,3] **n_estimators:** [100,200]	**GB**(0.1,1,200)	0.34	94.59 ± 3.9	92.65 ± 4.1	95.27 ± 3.7	93.28 ± 3.9	93.82 ± 4.4	91.91 ± 4.7	94.52 ± 4.0	93.57 ± 4.1	98.24 ± 1.9	94.65 ± 4.1
SVM(C,gamma, kernel)	**C:** [0.1,1,10,100] **gamma:** [0.1,1] **kernel:**[‘linear’,‘poly’, ‘rbf’, ‘sigmoid’]	**SVM** (10,1,‘rbf’)	0.17	94.65 ± 3.9	95.38 ± 4.1	95.75 ± 3.6	93.73 ± 3.8	93.44 ± 4.8	94.50 ± 5.1	94.55 ± 4.1	94.57 ± 4.2	98.21 ± 1.9	95.69 ± 4.1
MLP (activation, alpha, hidden_layer_sizes, learning_rate)	**Activation:**[‘tanh’, ‘relu’] **alpha:**[0.05, 0.1, 0.5] **hidden_layer_sizes:** [(50,50),(50,100,50),(100,)] **learning_rate:**[‘constant’, ‘adaptive’]	**MLP**(‘tanh’,0.05, (50,100,50), ‘adaptive’)	0.32	92.90 ± 4.6	91.79 ± 4.4	93.37 ± 4.8	92.50 ± 4.8	92.58 ± 5.0	92.06 ± 5.2	92.86 ± 4.7	92.71 ± 4.5	97.56 ± 2.5	93.79 ± 4.4

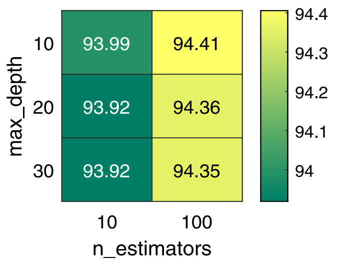


Fig. 10. Performance (F-score) variation on changing 'max_depth' and 'n_estimators' parameters for Random Forest.

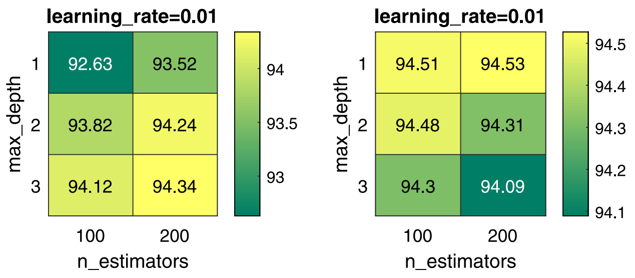


Fig. 11. Varying 'max_depth' and 'n_estimators' parameters for different learning rates of Gradient Boosted.

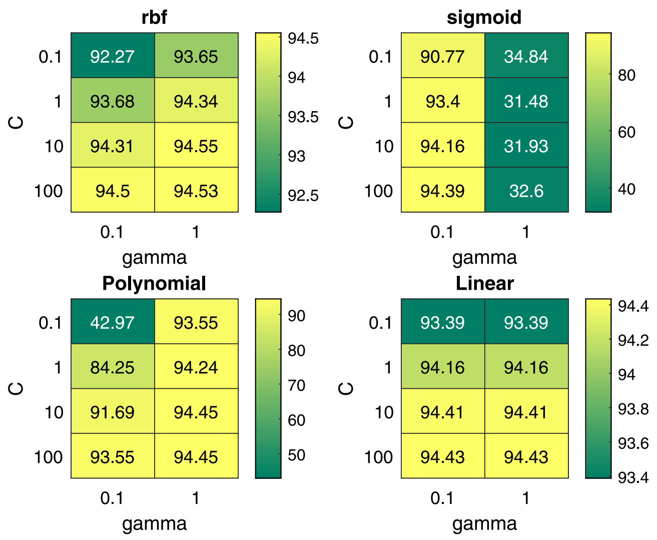


Fig. 12. Performance (F-score) variation on changing 'C' and 'gamma' parameters for different SVM kernels.

5.4. Discussion

The driving motivation for this research work is to analyze the efficacy of various textual features and machine learning classification algorithms for the task of authorship verification. The task is to assess which features and algorithm as well as what set of fine tuned parameters help achieve better results. From Table 5 it is evident that SVM outperformed other classifiers in terms of different performance parameters. Moreover, SVM with 'rbf' kernel took lowest time to train. Tabulated values represent the average scores of all users. In order to demonstrate the performance of each fine tuned classifier for all users, a boxplot (Fig. 9) is shown with 25 to 75% interquartile range. It is seen that for each classifier most of the values skewed to the higher Fscore range having a median of 95% (with an exception to MLP). For some 25% users, Fscore values ranged from lower quartile (approx 92%) to the minimum whisker of 83%. Only a small amount of users obtained F-score less than 82%–83%. In case of MLP, variation in number of neurons and hidden layers was not performed, hence, performance varied a bit differently for the best encountered settings of the MLP classifier. Doing so may help improve results for neural network as well.

Also, it is observed that every respective fine-tuned classifier attained a good and almost similar performance on their respective setting of parameters. But model training time varies with the parameter settings of each classifier. As per Occam's razor test [58], if different models have comparable performance, then simpler one should be picked.It is evident from the experiments that for the undertaken problem, choice of classifier does not produce much difference in performance as well as model building but the corresponding parameters in each classifier do. Effect of varying the parameters for each classifier can be analyzed from Figs. 10–13.

- From the experimental analysis, it is seen that choice of *n_neighbors* in kNN does not carry much difference which indeed signifies that the data of two classes have significant differences easily determined by any number of neighbors and hence, inclines the unknown sample towards its respective class. Experiments are performed varying the values of *n_neighbors* from 3 to 9, but the accuracy difference of merely 0.4% is seen in results.
- Similarly, literature reveals that Random Forest gives good performance at its default settings which are RF(*None, sqrt(number of features), 10*). But it is also believed that performance of every classifier and parameters is data dependent. Therefore, we varied the parameters in steps and found that accuracy slightly improved on increasing *max_depth* and *n_estimators* (Fig. 10). With the increase in *n_estimators* and *max_depth*, computational time for training and validation increased highly only to add a negligible difference in performance improvement(0.43%). Hence, varying the parameters does not seem a suitable choice for Random Forest.
- In the other ensemble method named Gradient Boosted, *learning_rate, max_depth* and *n_estimators* are incrementally increased. As in Fig. 11, it is observed that increasing the learning_rate improved the performance only marginally whereas change in other parameters such as *max_depth* and *n_estimators* portray considerable improvement.
- Likewise, in SVM, *C, gamma* and *kernel* parameters are varied and it is observed that the choice of kernel complimented with the C and gamma parameters impact the performance. As evident from Fig. 12, sigmoid kernel has a deteriorated performance (about 60% fall) with high values of gamma but on the other hand, with a fixed gamma value (0.1) and increased values of C, around 4% improvement in performance is observed. From C = 10 to C = 100 only 0.2% improvement is seen, hence, in between values of C as well as values beyond 100 were not preferred to be chosen. With *rbf* and *poly* kernels, performance increased up to a certain limit with increase in both C and gamma. In case of poly kernel a stable performance (94.57%) is achieved with values of C and gamma to be 1. Similarly, in rbf kernel, after C = 10 and gamma = 1, there is a slight decrease in performance. Overall rbf kernel with C and gamma values set to 10 and 1 respectively attained 94.55% F-score which is better than the other combinations.
- Similar to the other classifiers, Multi layer Perceptron (MLP) involves the setting of *alpha, activation function,* number and size of *hidden layers* and *learning_rate*. Only

Table 6
Comparative analysis of TB-CoAuth with existing techniques.

Technique	Features	Classifier	Accuracy (Avg)(%)	Precision (Avg)(%)	Recall (Avg)(%)	F-score (Avg)(%)	ROC_AUC (Avg)(%)
TB-CoAuth	Content-specific and Content-free	SVM with 'rbf' kernel	94.38	93.73	94.50	94.57	95.69
Barbon et al. [14]	char n-grams (SPI measure)	kNN	88.36	88.72	87.78	88.22	92.81
Li et al. [12]	Stylometric Features	Voting Algorithms (Ensemble)	84.57	84.24	84.87	84.52	84.57
Li et al. [12]	Stylometric Features	Decision Trees	70.58	70.35	70.59	70.43	70.59
Brocardo et al. [7]	n-grams	Hybrid SVM-LR	89.57	89.59	89.54	89.53	95.22

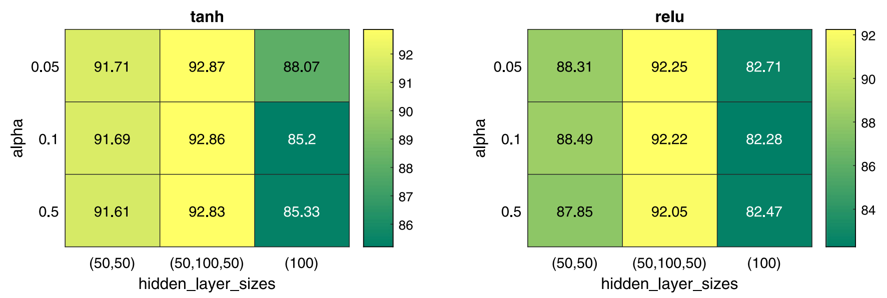


Fig. 13. Performance (F-score) variation on changing 'alpha' and 'hidden_layer_sizes' for 'tanh' and 'relu' activation function of MLP with constant learning rate.

two activation functions namely, *tanh* and *relu* were used out of which *tanh* performed comparatively better attaining a maximum and minimum F-score of 92.86% and 85.12% respectively. Among other parameters choice of alpha and hidden layers affect the performance most. It is observed from Fig. 13 that increasing alpha only had a marginal impact on performance but with the increase in number of hidden layers performance increased by 7% (from 85.12% F-score to 92.86%). Number of neurons in each hidden layer are not varied much because of the computational resources and time constraints.

5.5. Comparative analysis

In order to compare the performance of the proposed TB-CoAuth method against existing methods, three baseline techniques Barbon et al. [14], Li et al. [12] and Brocardo et al. [7] which have also worked on the detection of compromised accounts in social networks using textual features have been studied and implemented. For a fair comparison, features and methods used in these existing techniques have been deployed on the collected Twitter dataset. Table 6 presents the tabulated performance comparison of TB-CoAuth against these techniques.

From Table 6 it is evident that TB-CoAuth outperformed other techniques providing 6.3%, 10.1%, 24.1% and 5.04% better average F-score than Barbon et al. [14], Li et al. [12] [Voting Algorithm Approach], Li et al. [12] [Decision Tree Approach] and Brocardo et al. [7] respectively.

6. Conclusion

In this work, efficiency of different textual features for the task of authorship verification and thereby the detection of compromised accounts in social networks have been examined. Experiments have been performed on tweets of a popular social network Twitter but it nowhere makes the proposed method specific to Twitter platform, indeed it could be deployed on any social network. Various feature selection techniques are applied to rank the features independently which are then aggregated using a popular rank aggregation technique called BORDA. Efficiency of various machine learning classifiers is also examined with experimental results stating that SVM with *rbf* kernel outperformed other classifiers namely, kNN, Random Forest, Gradient Boosted and Multi Layer Perceptron, attaining a maximum F-score of 94.57% under the varied parameter settings. But overall, all the classifiers performed equally well with above 92% average accuracy and F-score. As the work undertaken is grounded only on text mining, hence, only text is deployed for continuous authentication which makes it an easy and unobtrusive approach to deploy at back end. In the near future, proposed work will be supplemented with some meta data information such as time, language and source to compare the effectiveness of plain textual information with the other alternatives. Secondly, availability of ground truth data for the undertaken problem is also a tedious and crucial task. Rather than working with the binary data and hence study problem as two class classification problem, one class and PU Learning classification approaches could be used that nullify the availability of negative class data. This may help portray more confidence in results and easy deployment of the technique with no boundations of negative class, definition and scope of which is always unclear in case of problems such as detection of compromised accounts.

CRediT authorship contribution statement

Ravneet Kaur: Conceptualization, Methodology, Formal analysis, Investigation, Data curation, Writing - original draft, Writing - review & editing, Visualization. **Sarbjeet Singh:** Supervision, Writing - review & editing, Validation, Visualization, Funding acquisition. **Harish Kumar:** Supervision, Writing - review & editing, Resources, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This publication is an outcome of the R&D work under the Visvesvaraya PhD Scheme of Ministry of Electronics & Information Technology, Government of India, being implemented by Digital India Corporation under the Grant Number: PhD/MLA/4(61)/2015-16.

References

[1] K. Olmstead, A. Smith, Americans and cybersecurity, Pew Res. Center 26 (2017) 1–43.

[2] C. Grier, K. Thomas, V. Paxson, M. Zhang, @spam: the underground on 140 characters or less, in: Proceedings of the 17th ACM Conference on Computer and Communications Security, ACM, 2010, pp. 27–37.

[3] H. Gao, J. Hu, C. Wilson, Z. Li, Y. Chen, B.Y. Zhao, Detecting and characterizing social spam campaigns, in: Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, ACM, 2010, pp. 35–47.

[4] E. Zangerle, G. Specht, Sorry, I was hacked: A classification of compromised twitter accounts, in: Proceedings of the 29th Annual ACM Symposium on Applied Computing, ACM, 2014, pp. 587–593.

[5] C. VanDam, J. Tang, P.-N. Tan, Understanding compromised accounts on twitter, in: Proceedings of the International Conference on Web Intelligence, ACM, 2017, pp. 737–744.

[6] R. Zheng, J. Li, H. Chen, Z. Huang, A framework for authorship identification of online messages: Writing-style features and classification techniques, J. Am. Soc. Inf. Sci. Technol. 57 (3) (2006) 378–393.

[7] M.L. Brocardo, I. Traore, I. Woungang, Authorship verification of e-mail and tweet messages applied for continuous authentication, J. Comput. System Sci. 81 (8) (2015) 1429–1440.

[8] M. Kocher, J. Savoy, A simple and efficient algorithm for authorship verification, J. Assoc. Inf. Sci. Technol. 68 (1) (2017) 259–269.

[9] M.L. Brocardo, I. Traore, I. Woungang, M.S. Obaidat, Authorship verification using deep belief network systems, Int. J. Commun. Syst. 30 (12) (2017) 1–10.

[10] M.L. Brocardo, I. Traore, I. Woungang, Continuous authentication using writing style, in: Biometric-Based Physical and Cybersecurity Systems, Springer, 2019, pp. 211–232.

[11] M.L. Brocardo, I. Traore, Continuous authentication using micro-messages, in: Privacy, Security and Trust (PST), 2014 Twelfth Annual International Conference on, IEEE, 2014, pp. 179–188.

[12] J.S. Li, L.-C. Chen, J.V. Monaco, P. Singh, C.C. Tappert, A comparison of classifiers and features for authorship authentication of social networking messages, Concurr. Comput.: Pract. Exper. 29 (14) (2016) 1–15.

[13] J. Peng, K.-K.R. Choo, H. Ashman, Bit-level n-gram based forensic authorship analysis on social media: Identifying individuals from linguistic profiles, J. Netw. Comput. Appl. 70 (2016) 171–182.

[14] S. Barbon, R.A. Igawa, B.B. Zarpelao, Authorship verification applied to detection of compromised accounts on online social networks, Multimedia Tools Appl. 76 (3) (2017) 3213–3233.

[15] R. Kaur, S. Singh, H. Kumar, AuthCom: Authorship verification and compromised account detection in online social networks using AHP-TOPSIS embedded profiling based technique, Expert Syst. Appl. 113 (2018) 397–414.

[16] C. VanDam, P.-N. Tan, J. Tang, H. Karimi, CADET: A multi-view learning framework for compromised account detection on twitter, in: 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM, IEEE, 2018, pp. 471–478.

[17] H. Karimi, C. VanDam, L. Ye, J. Tang, End-to-end compromised account detection, in: 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM, IEEE, 2018, pp. 314–321.

[18] R. Kaur, S. Singh, H. Kumar, Authorship analysis of online social media content, in: Proceedings of 2nd International Conference on Communication, Computing and Networking, Springer, 2019, pp. 539–549.

[19] A. Abbasi, H. Chen, Writeprints: A stylometric approach to identity-level identification and similarity detection in cyberspace, ACM Trans. Inf. Syst. 26 (2) (2008) 7–36.

[20] A. Rocha, W.J. Scheirer, C.W. Forstall, T. Cavalcante, A. Theophilo, B. Shen, A.R. Carvalho, E. Stamatatos, Authorship attribution for social media forensics, IEEE Trans. Inf. Forensics Secur. 12 (1) (2017) 5–33.

[21] F. Iqbal, L.A. Khan, B. Fung, M. Debbabi, E-mail authorship verification for forensic investigation, in: Proceedings of the 2010 ACM Symposium on Applied Computing, ACM, 2010, pp. 1591–1598.

[22] S. Argamon, M. Koppel, J.W. Pennebaker, J. Schler, Automatically profiling the author of an anonymous text, Commun. ACM 52 (2) (2009) 119–123.

[23] M. Fatima, K. Hasan, S. Anwar, R.M.A. Nawab, Multilingual author profiling on Facebook, Inf. Process. Manage. 53 (4) (2017) 886–904.

[24] N. Potha, E. Stamatatos, Improved algorithms for extrinsic author verification, Knowl. Inf. Syst. (2019) 1–19.

[25] A.B. Duque, F.d.A. de Carvalho, R. Vimieiro, A multiview clustering approach for mining authorial affinities in literary texts, in: 2019 8th Brazilian Conference on Intelligent Systems, BRACIS, IEEE, 2019, pp. 818–823.

[26] E. Stamatatos, Plagiarism detection using stopword n-grams, J. Am. Soc. Inf. Sci. Technol. 62 (12) (2011) 2512–2527.

[27] M. AlSallal, R. Iqbal, V. Palade, S. Amin, V. Chang, An integrated approach for intrinsic plagiarism detection, Future Gener. Comput. Syst. 96 (2019) 700–712.

[28] A. Rocha, W.J. Scheirer, C.W. Forstall, T. Cavalcante, A. Theophilo, B. Shen, A.R. Carvalho, E. Stamatatos, Authorship attribution for social media forensics, IEEE Trans. Inf. Forensics Secur. 12 (1) (2016) 5–33.

[29] M. Potthast, P. Rosso, E. Stamatatos, B. Stein, A decade of shared tasks in digital text forensics at PAN, in: European Conference on Information Retrieval, Springer, 2019, pp. 291–300.

[30] F. Iqbal, H. Binsalleeh, B.C. Fung, M. Debbabi, A unified data mining solution for authorship analysis in anonymous textual communications, Inform. Sci. 231 (2013) 98–112.

[31] R.A. Igawa, A. Almeida, B. Zarpelão, S. Barbon Jr, Recognition on Online Social Network by user's writing style, iSys-Rev. Bras. Sistemas de Inf. 8 (3) (2016) 64–85.

[32] B. Viswanath, M.A. Bashir, M. Crovella, S. Guha, K.P. Gummadi, B. Krishnamurthy, A. Mislove, Towards detecting anomalous user behavior in online social networks, in: USENIX Security Symposium, 2014, pp. 223–238.

[33] M. Koppel, Y. Winter, Determining if two documents are written by the same author, J. Assoc. Inf. Sci. Technol. 65 (1) (2014) 178–187.

[34] R.M. Green, J.W. Sheppard, Comparing frequency-and style-based features for twitter author identification, in: FLAIRS Conference, AAAI, 2013, pp. 64–69.

[35] S. Seidman, Authorship verification using the impostors method, in: CLEF 2013 Evaluation Labs and Workshop-Online Working Notes, Citeseer, 2013.

[36] O. Halvani, M. Steinebach, VEBAV-a simple, scalable and fast authorship verification scheme, in: CLEF (Working Notes), 2014, pp. 1049–1062.

[37] T. Neal, K. Sundararajan, D. Woodard, Exploiting linguistic style as a cognitive biometric for continuous verification, in: 2018 International Conference on Biometrics, ICB, IEEE, 2018, pp. 270–276.

[38] A. Salem, A. Almarimi, G. Andrejková, Text dissimilarities predictions using convolutional neural networks and clustering, in: 2018 World Symposium on Digital Intelligence for Systems and Machines, DISA, IEEE, 2018, pp. 343–347.

[39] E. Stamatatos, A survey of modern authorship attribution methods, J. Am. Soc. Inf. Sci. Technol. 60 (3) (2009) 538–556.

[40] R. Kaur, S. Singh, H. Kumar, Rise of spam and compromised accounts in online social networks: A state-of-the-art review of different combating approaches, J. Netw. Comput. Appl. 112 (2018) 53–88.

[41] D. Seyler, L. Li, C. Zhai, Identifying compromised accounts on social media using statistical text analysis, 2018, arXiv preprint arXiv:1804.07247.

[42] D. Trång, F. Johansson, M. Rosell, Evaluating algorithms for detection of compromised social media user accounts, in: 2015 Second European Network Intelligence Conference, IEEE, 2015, pp. 75–82.

[43] M. Kocher, J. Savoy, UniNE at CLEF 2016: Author profiling, in: CLEF (Working Notes), 2016, pp. 903–911.

[44] G. Salton, C. Buckley, Term-weighting approaches in automatic text retrieval, Inf. Process. Manage. 24 (5) (1988) 513–523.

[45] J.C. de Borda, Mémoire sur les élections au scrutin, Histoire de lÁcademie Royale des Sciences, 1781.

[46] L.H. Lorena, A.C. Carvalho, A.C. Lorena, Filter feature selection for one-class classification, J. Intell. Robot. Syst. 80 (1) (2015) 227–243.

[47] E. Namsrai, T. Munkhdalai, M. Li, J.-H. Shin, O.-E. Namsrai, K.H. Ryu, A feature selection-based ensemble method for arrhythmia classification, J. Inf. Process. Syst. 9 (1) (2013) 31–40.

[48] Q. Shen, R. Diao, P. Su, Feature selection ensemble, Turing-100 10 (2012) 289–306.

[49] R. Wald, T.M. Khoshgoftaar, D. Dittman, W. Awada, A. Napolitano, An extensive comparison of feature ranking aggregation techniques in bioinformatics, in: Information Reuse and Integration (IRI), 2012 IEEE 13th International Conference on, IEEE, 2012, pp. 377–384.

[50] A. Tsymbal, M. Pechenizkiy, P. Cunningham, Diversity in search strategies for ensemble feature selection, Inf. Fusion 6 (1) (2005) 83–98.

[51] R.C. Prati, Combining feature ranking algorithms through rank aggregation, in: Neural Networks (IJCNN), the 2012 International Joint Conference on, IEEE, 2012, pp. 1–8.

[52] C. Dwork, R. Kumar, M. Naor, D. Sivakumar, Rank aggregation methods for the web, in: Proceedings of the 10th International Conference on World Wide Web, ACM, 2001, pp. 613–622.

[53] A. Tsymbal, M. Pechenizkiy, P. Cunningham, Diversity in Ensemble Feature Selection, Technical Report TCD-CS-2003-44, The University of Dublin, 2003.

[54] M. Al-Ayyoub, A. Alwajeeh, I. Hmeidi, An extensive study of authorship authentication of arabic articles, Int. J. Web Inf. Syst. 13 (1) (2017) 85–104.

[55] M. Al-Ayyoub, Y. Jararweh, A. Rabab'ah, M. Aldwairi, Feature extraction and selection for Arabic tweets authorship authentication, J. Ambient Intell. Humanized Comput. 8 (3) (2017) 383–393.
[56] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in python, J. Mach. Learn. Res. 12 (2011) 2825–2830.
[57] R. Li, S. Wang, H. Deng, R. Wang, K.C.-C. Chang, Towards social user profiling: unified and discriminative influence model for inferring home locations, in: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2012, pp. 1023–1031.
[58] A. Blumer, A. Ehrenfeucht, D. Haussler, M.K. Warmuth, Occam's razor, Inf. Process. Lett. 24 (6) (1987) 377–380.