IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 28, 2020, accepted August 6, 2020, date of publication August 14, 2020, date of current version August 26, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3016664

Do Many Models Make Light Work? Evaluating Ensemble Solutions for Improved Rumor Detection

YOUNGHWAN KIM[1], HUY KANG KIM[1], (Member, IEEE), HYOUNGSHICK KIM[2,3], (Associate Member, IEEE), AND JIN B. HONG[4], (Member, IEEE)

[1]School of Cybersecurity, Korea University, Seoul 02841, South Korea
[2]Department of Computer Science and Engineering, College of Software, Sungkyunkwan University, Seoul 16419, South Korea
[3]CSIRO Data61, Eveleigh, NSW 2015, Australia
[4]Department of Computer Science and Software Engineering, The University of Western Australia, Crawley, WA 6009, Australia

Corresponding author: Jin B. Hong (jin.hong@uwa.edu.au)

This work was supported in part by the Australia-Korea Foundation (AKF) of the Department of Foreign Affairs and Trade (DFAT) project Intelligent Cybersecurity: Jumpstarting Collaboration on Smarter Security for Dynamic Networks, and in part by the framework of international cooperation program managed by the National Research Foundation of Korea under Grant NRF-2017K1A3A1A17092614.

ABSTRACT There have been many efforts to detect rumors using various machine learning (ML) models, but there is still a lack of understanding of their performance against different rumor topics and available features, resulting in a significant performance degrade against completely new and unseen (unknown) rumors. To address this issue, we investigate the relationship between ML models, features, and rumor topics to select the best rumor detection model under specific conditions using 13 different ML models. Our experiment results demonstrate that there is no clear winner among the ML models in all different rumor topics with respect to the detection performance. To overcome this problem, a possible way is to use an ensemble of ML models. Although previous work presents an improved detection of rumors using ensemble solutions (ES), their evaluation did not consider detecting unknown rumors. Further, they did not present nor evaluate the configuration of the ES to ensure that it indeed performs better than using a single ML model. Based on these observations, we propose to evaluate the use of an ES by examining their unknown rumor detection performance compared with single ML models but as well as different configurations of the ESes. Our experimental results using real-world datasets found that an ES of Random Forest, XGBoost and Multilayer perceptron overall produced the best F1 score of 0.79 for detecting unknown rumors, a significant improvement compared with a single best ML model which only achieved a 0.58 F1 score. We also showed that not all ESes are the same, with significantly degraded detection and large variations in performance when different ML models are used to construct the ES. Hence, it is infeasible to rely on any single ML model-based rumor detector. Finally, our solution also performed better than other recent detectors, such as eventAI and NileTMRG that performed similar to using a single ML model – making it a much more attractive solution to detect unknown rumors in practice.

INDEX TERMS Feature analysis, machine learning, rumor detection, social media, ensemble solution, Twitter.

I. INTRODUCTION

The spread of rumors is still prevalent today. To address this issue, various machine learning (ML) models have been proposed to detect rumors [1], [2]. The majority has focused on improving the performance of ML models for specific rumor events (e.g., crime-related tweets from [3]), but mostly they were applied in the post-analysis phase (i.e., after the rumor has spread/collected). Consequently, many ML models were able to achieve a competitive performance (in terms of accuracy, F1 scores, etc.), but using them to detect new and unseen (unknown) rumor events significantly degraded their detection performance (which we demonstrate later in Section IV). For example, there was a large number of rumors

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

and fake news during the 2016 U.S. presidential election [4], such as ''Trump had won both the popular vote and the Electoral College''. In particular, the term *unknown rumors* in this paper refers to those rumors that have not been seen previously nor fact-checked. In addition, available features can also significantly affect the performance of rumor detection [5]. Consequently, retraining those ML models with a new dataset every time to maintain the performance is impractical. Hence, it is crucial to develop a solution that maintains high detection performance under various environmental conditions, such as detecting unknown rumor events that yield different features.

To address the aforementioned problems, a better rumor detection solution is needed, which takes into account various environmental conditions. As a first step, we analyze the relationship between the rumor topics, features and ML models to understand the factors that affect their rumor detection performances, particularly for new and previously unforeseen rumors (here, a rumor event is an instance of the rumor topic, and we will denote new and unforeseen ones as *unknown rumors*). To achieve this goal, we conduct several experiments using two publicly available datasets: (1) *PHEME* [6], and (2) RumourEval2019 (RE2019)[1] [7]. Both datasets served as ground truth in previous studies such as in [8] and [7], so that our results can be compared easily with rumor detectors. Next, rumor features are collected such as in [9], which are categorized into three *feature groups*. Finally, we implement 13 ML models to detect rumors with respect to feature groups and compare their performances. The results indicate that there is no clear winner, and the rumor detection performance differs significantly when rumor events and available features are changed.

To improve unknown rumor detection, we develop and evaluate several ensemble solutions (ES) by combining multiple ML models in different ways. Although it is a widely accepted fact that using ESes improves the performance over single ML models, to the best of our knowledge, there are no (or very limited number of) studies that evaluate the detection of unknown rumors. Hence, although many frameworks, including various ESes, have been used previously for rumor detection [10], [11], they did not provide methods to construct ESes that would be effective in detecting unknown rumors, and they also lacked in systematic evaluation of the performance factors with respect to rumor topics, ML models and available features. Through experimental analysis using real-world datasets, we show that indeed randomly constructing the ES cannot detect unknown rumors (and even known ones) sufficiently and often worse than using a single ML model. Therefore, it is infeasible to assume that using *any* ML models to construct the ES would improve rumor detection, especially dealing with unknown rumors.

[1]RE2019 is mainly used for rumor verification, but this dataset still contains annotated rumor data that can be used for the detection stage (i.e., ignore all annotations and use raw data for detection).

In this paper, we propose an approach for optimizing ES configurations to address the aforementioned problems. In particular, we focus on detecting rumors, the first stage in rumor classification [12].[2] Through our experimental analysis, we found that using a well-formulated ES (e.g., ES combining Random Forest, XGBoost, and Multilayer perceptron denoted as RXM-ES) achieved 0.79 F1-score, compared to 0.58 using a single ML model and 0.62 using random ES model for detecting unknown rumors. Similarly, the RXM-ES achieved an average F1 score of 0.62, compared with 0.51 using a single ML model in our 10-fold cross-validation using the RE2019 dataset. These results indicate that not all ESes could detect unknown rumors the same, and therefore requires a careful selection of ML models to construct the ES (such as RXM-ES in our experiment). Hence, appropriately selecting ML models to construct the ES can significantly improve the detection of unknown rumors compared to existing solutions (particularly single-ML-based models), which is an essential step for stance and veracity classifications to improve the analysis of chosen rumors and their context [12]. The contributions of our paper are summarized as follows.

- To segregate ML models based on the performance with varying features and/or rumor topics;
- To rank ML models for detecting rumors with respect to rumor topics and available features;
- To compare different configurations of the ensemble solutions using different ML models to detect unknown rumors;
- To perform experimental analysis to evaluate the performance of different ML models, ESes, and existing rumor detection tools (e.g., eventAI and NileTMRG) using two public datasets.

The rest of the paper is organized as follows. Section II presents the related work on rumor detection. Section III presents the overall process of evaluating ML models for rumor detection. Section IV presents the experimental results with various ML models and discusses the results. Section V presents the ES formulation, with experimental results in Section VI. Section VII discusses our findings and limitations, then finally, Section VIII concludes this paper.

II. RELATED WORK

Various ML models have extensively been studied in the past decade to enhance their rumor detection capabilities [7], [12], [13]. For instance, Yang *et al.* [14] identified rumors on Twitter using Logistic Regression (LR), Naive Bayes (NB), and Random Forest (RF) based on hot topic detection. Zhao *et al.* [15] presented enquiry-based rumor detection, but their result reported the performance using precision only, which only achieved 0.52. Further, the solution relies on the existence of enquiries for the rumor event (i.e., if there are no sufficient queries about the rumor, it may not be detected).

[2]Rumor tracking, stance classification, and veracity classification are out of scope in this paper.

Lin *et al*. [16] proposed early rumor detection based on user attitude using convolutions neural network and BERT neural network language model. Although results show effective early detection, it requires legitimate users to comment about the addressed topic, which can be targeted by bots [17].

Although performance improvement was observed, the performance of existing ML techniques is significantly degraded for unseen and unexpected rumor events (see Section IV). Other approaches based on stances require legitimate user interactions, which may not be practical for all events shared in social media platforms. To overcome these limitations, we analyze the relationships between rumor topics, features and detection models, and suggest an ensemble solution using multi-ML models to improve the detection performance (see Section V).

A. ENSEMBLE SOLUTIONS FOR RUMOR DETECTION

Dietterich *et al.* [18] explained that when using an ensemble method rather than looking for a single highest performing algorithm model, it is more advantageous for statistical, computational, and representative issues. In addition, empirical studies in several previous papers have confirmed that the use of the ensemble solution can reduce high variability, variance, and bias [19]–[21]. To grasp these benefits, there have been several approaches to use ensemble solutions to detect rumors. Liu *et al.* [10] proposed an ensemble learning approach that combines different data sampling methods (i.e., random oversampling, random undersampling and fuzzy-based oversampling). However, their main application is spam detection, and does not consider the use of different ML models. Nguyen *et al.* [11] presented an ensemble solution based on features (i.e., the credibility of each tweet and the credibility of overall events) for modeling tweet-level credibility. They showed that the proposed model achieves over 80% accuracy in the first hour, going up to over 90% accuracy over time. However, they did not consider how the performance changes when used against different rumor topics and/or events. Geng *et al.* [22] approached through a multi-view neural network model with respect to content, reply and sentiment. However, those existing ES-based approaches do not comprehensively evaluate their rumor detection performances with respect to varying rumor topics and available features. Moreover, there is no suggestions or discussion on how to construct the ES.

B. RUMOR DETECTION FEATURES

There are numerous features that can characterize rumors, such as temporal [23], structural [24], message [25], network [25], user [26], content [26], etc. However, using more features does not necessarily improve the detection performance [25]. Castillo *et al.* [9] identified classes of features associated with rumors, and demonstrated that context and propagation features are specifically effective for rumor detection, indicating the importance of understanding the context of the rumor and the graph patterns in the social network. Kwon *et al.* [23] proposed a rumor detection method using temporal, linguistics, and structural features on Twitter. Using their proposed features, they were able to achieve up to 0.89 F1 score, indicating the importance of using appropriate features. Kwon *et al.* [27] analyzed the relative importance of user, structural, linguistic, and temporal features for rumor classification on Twitter. Based on the results, they suggested a new rumor classification algorithm that achieves competitive accuracy over both short and long time windows. Ma *et al.* [28] used a time series model to capture temporal characteristics of rumors, achieving up to 0.89 F1 score. However, other features are not compared in their work. Giasemidis *et al.* [25] extracted 87 features classified into three categories of message-based, user-based, and network-based in a dataset containing 72 rumors on Twitter, and conducted a study to identify rumors using various machine learning algorithms. The best F1 score achieved was using the Decision Tree (DT) with the value 0.97. Although various features are explored to improve the performance of rumor detection models, previous work did not take into account how different features performed when they are used for other rumor topics or events. In addition, the ability to collect some of those features is not feasible until a certain amount of rumor has spread, limiting their usage in rumor detection.

C. DEEP LEARNING FOR RUMOR DETECTION

Another rising technique is to use deep learning (DL). Guo *et al.* [29] presented a hierarchical LSTM network with social attention, achieving an accuracy of 0.84 and F1 score of 0.83 using a Twitter dataset. Chen *et al.* [30] presented a deep attention model built on RNNs, which selectively learns temporal representations of sequential posts to identify rumors. Wang *et al.* [5] presented a Neural Model using Dynamic Propagation Structures. They also presented the performance variation with different features, which showed a significant difference with the highest F1 score of 0.83 with content and structure features combined. Ruchansky *et al.* [31] presented a hybrid DL model *CSI* using LSTM to capture the temporal pattern of user activity and RNN to characterize the user behavior to detect fake news on Twitter and Weibo. Ma *et al.* [32] presented generative adversarial learning for rumor detection, which strengthened the discriminator to learn stronger rumor indicative representations.

D. DEEP LEARNING LIMITATIONS

Although DL-based models showed a performance improvement in rumor detection, previous work did not evaluate their model's performance when used for detecting unknown rumors [13], [30].[3] Further, DL requires a large dataset for training [33], as well as LSTM-based models requiring sequential data. Hence, these approaches may not be

[3]There are many works on unknown rumor veracity such as in [6], but the focus of this paper is on rumor detection, the first step when raw input is provided, whereas rumor veracity handles already identified rumors to confirm whether the rumor is true, debunked as false, or unverified.

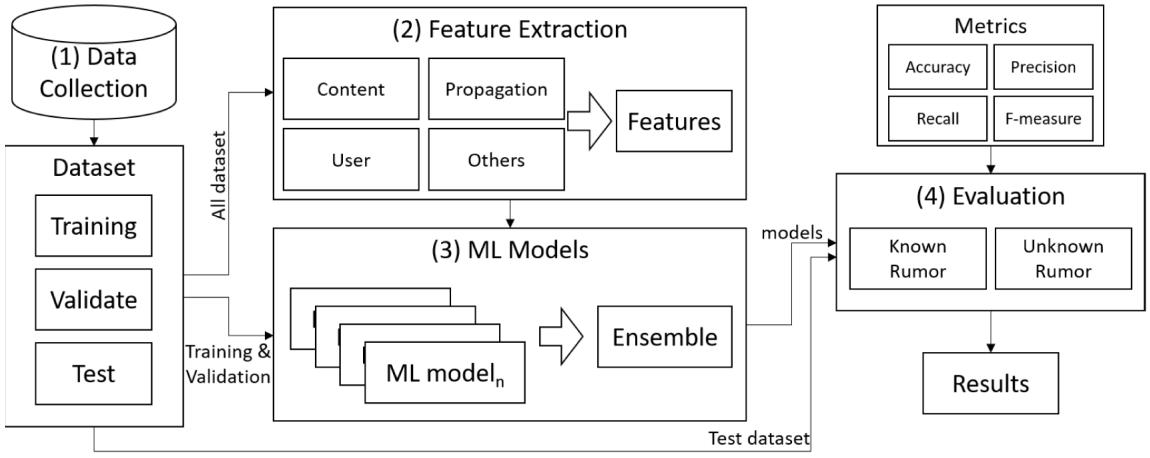


FIGURE 1. Proposed rumor detection framework for unknown rumors.

suitable when rumor features do not exhibit sequential properties (e.g., structural and user features). This is not ideal for early detection of unknown rumors where those topics may not be well represented as sequences of past rumor topics. To validate, we implemented an equivalent DS-based rumor detection model[4] presented in [31]. Using this model, the best results we obtained were accuracy of 0.56 and F1 score of 0.50 for detecting unknown rumor events (i.e., inputting rumor event data not used previously in training). Therefore, we conclude that the performance of all existing rumor detection models, including DL-based models, are significantly worse when used for detecting unknown rumors, which is a huge constraint in practice.

III. EVALUATION FRAMEWORK OF ML-BASED RUMOR DETECTION MODELS

Rumor detection is the first component in the rumor classification system as described by Zubiaga *et al.* [12]. The rumor detection component identifies whether a piece of information constitutes a rumor. The remaining components (i.e., rumor tracking, stance classification, and veracity classification) rely on correctly detected rumors from the rumor detection component. Therefore, it is of paramount importance to ensure the performance and accuracy of the rumor detection component are high. Rumor detection using ML models normally has four main steps: (1) data collection, (2) feature extraction, (3) ML model training, and (4) performance evaluation, as depicted in Figure 1. Further details of each step are described as follows.

Step 1 (Data Collection): Raw data can be collected from social media, such as Twitter, using various APIs, which are then preprocessed to extract features. In this paper, the data collection step is simplified using a publicly available dataset named *PHEME* [6] and *RumourEval2019* (denoted as RE2019) [7]. PHEME is a large-sized dataset containing nine distinctive events across five rumor topics from Twitter with 6,425 tweets. Due to the size of the dataset being small for "Ebola" (Medical) and "Gurlitt" (Political) events, we combined them to create a new dataset *Mixed* with multiple rumor topics. Table 1 shows the statistics of the labeled tweets for each event in the PHEME dataset.[5] RE2019 is a smaller sized dataset containing 27 distinctive events across two rumor topics (Political and Impact) from Twitter with 112 tweets (duplicates with PHEME removed, which was originally 381 tweets) and 52 individual Reddit posts across two rumor topics (Political and Impact). Existing rumor detection techniques can also be used to collect and classify rumor data. However, as we find out later in Section IV, using existing techniques, especially single ML model-based ones, may not be accurate to produce correctly labeled data due to their lack of performance against unknown rumor events.

Step 2 (Feature Extraction): Rumor features are categorized into three groups similar to the work in [25]: Content (C), Propagation (P), and User (U). There are many

TABLE 1. Classifying PHEME dataset rumor topics [6] (R: Rumor, NR: Non-Rumor).

Rumor Topic	Event	R	NR	Total
Crime	Sydney Siege	522	699	1,221
	Ottawa Shooting	470	420	890
	Ferguson	284	859	1,143
	Charlie Hebdo	458	1,621	2,079
Politic	Putin Missing	126	112	238
Entertainment	Prince	229	4	233
Impact	Germanwings Crash	238	231	469
Mixed	Ebola & Gurlitt	75	77	152
Total		2,402	4,023	6,425

[4]The implementation of the CSI-equivalent model can be found at https://github.com/hksecurity/RumorDetection-with-Multi-ML-Models.

[5]Rumor topics are populated from https://www.kaggle.com/sfarooqi/news-classification.

TABLE 2. Rumor feature groups with selected features.

Group	Features
Content (C)	has question mark
	has exclamation mark
	has URL
	has hashtag
	unigram bow vector
	POS tagging
Propagation (P)	time span
	retweet count
	favorite count
User (U)	is verified
	has description
	followers count
	friends count
	statuses count

other features, but not all of them are useful for detecting rumors. For example, Castillo *et al*. [9] showed useful features for detecting rumors on Twitter. Based on their findings, we selected the top 14 features to be used in this paper. Table 2 summarizes the rumor features into three feature groups.

Step 3 (ML Models and Training): ML models can be categorized into five classes [34]: (1) logic-based (e.g., Decision Tree), (2) perceptron-based (e.g., Multilayer Perceptron), (3) Statistical learning (e.g., Naive Bayes), (4) instance-based (e.g., K-Nearest Neighbor), and (5) support vector machines (e.g., Linear and RBF SVM). We selected 13 most popular ML models covering all ML categories, and implemented them for comparisons using scikit-learn library [35]. They are (of no particular order): Logistic Regression (LR), Decision Tree (DT), K-Nearest Neighbor (KNN), Support Vector Machine (SVM), Gaussian Process (GP), Random Forest (RF), Naive Bayes (NB), Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), Gradient Boosting (GBM), Adaptive Boost (AdaBoost), XGBoost, and Multilayer perceptron (MLP). Accuracy, Precision, Recall, and F1 score are used to measure the performance (see their definitions in [36]). In the context of rumor detection, they are represented as follows.

- **Accuracy**: the proportion of correctly classified tweets;
- **Precision**: the proportion of tweets classified as rumors that actually are rumors;
- **Recall**: the proportion of rumors that were accurately classified;
- **F-measure**: the harmonic mean of *precision* and *recall*.

In our experiment, we divide the dataset into three categories; *train*:*validate*:*test* for the train set, validation set, and test set, respectively with the ratio 2 : 2 : 6 (further details in Section VI). The training set is used to optimize the ML models' configurations. The validation set is used for the ensemble solutions to assign appropriate weight values to different ML models in the set (details are given later in Section V). Lastly, the test set is used to measure the performance for all ML models and the ensemble solutions.

TABLE 3. Best ML model for detecting known rumors.

Rumors	Features			
	All	C	P	U
Crime				
Sydney Siege	**NB** 0.84	NB **0.84**	MLP 0.75	RF 0.72
Ottawa Shooting	**GBM 0.92**	**GBM 0.92**	DT 0.70	RF 0.73
Ferguson	NB 0.85	**RF 0.87**	XGB 0.59	RF 0.59
Charlie Hebdo	**MLP 0.86**	RF 0.86	XGB 0.73	RF 0.70
Politic				
Putin Missing	RF 0.85	**MLP 0.88**	DT 0.78	LR 0.73
Entertainment				
Prince	**RF 0.83**	QDA 0.57	All 0.50	All 0.50
Impact				
German Wings	RF 0.96	**RF 0.97**	MLP 0.69	GBM 0.62
Mixed				
Ebola & Gurlitt	**LR 0.94**	**MLP 0.94**	MLP 0.71	DT 0.74

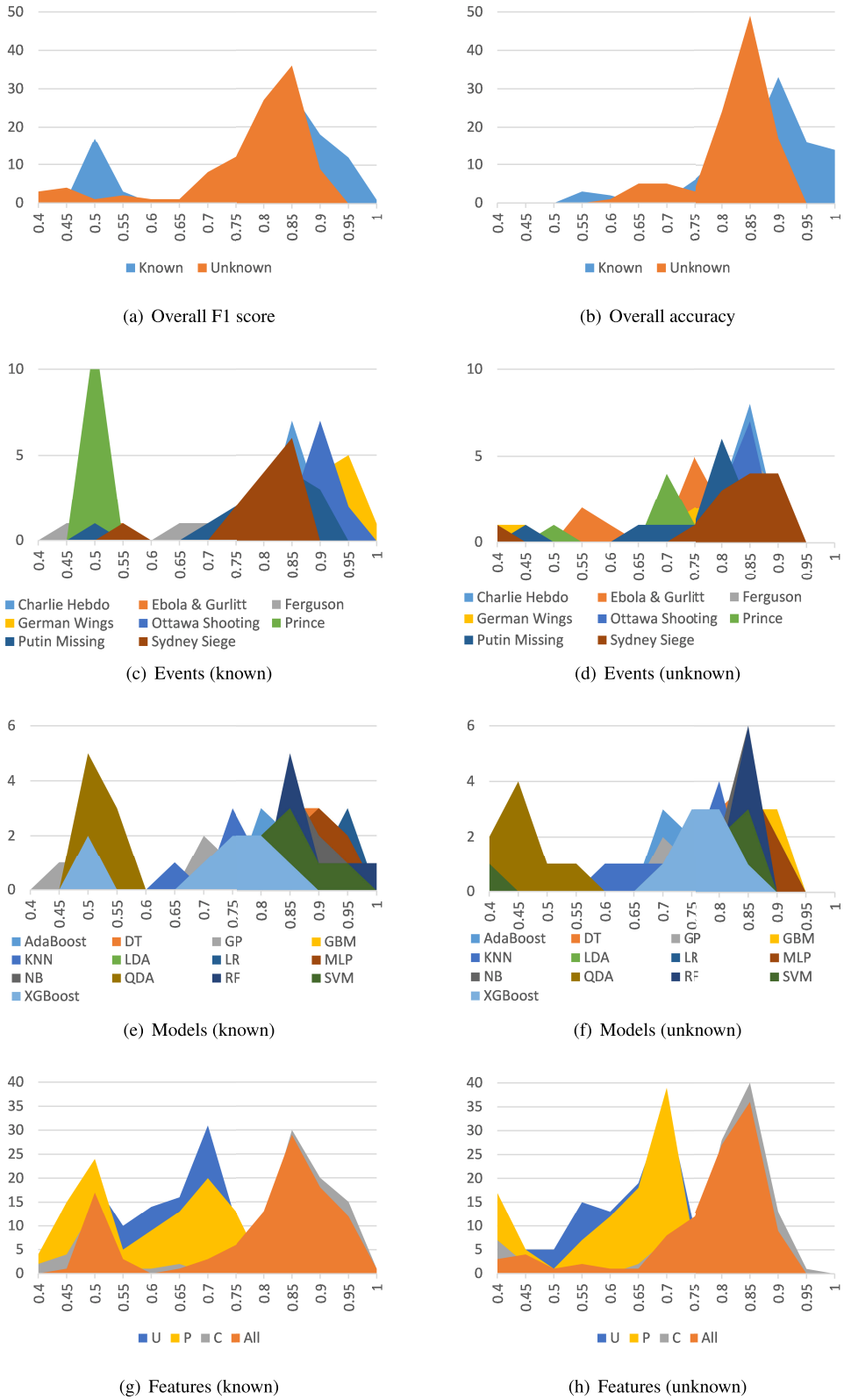
For testing the detection of known rumor events, we train the models using the rumor event data. This approach is the same as previous work [12], [13]. To test the detection of unknown rumor events, we train the models using *other* rumor event data (i.e., to test unknown rumor event *Ferguson*, the training data will consist of all other events but excluding *Ferguson* dataset), which in effect treats the testing rumor dataset as previously unseen.

Further, the Hyperopt-sklearn package utilizes a TPE (Tree-Parzen Estimator) algorithm to find parameter values that are likely to derive high performance in a predefined hyperparameter space. Therefore, we perform hyperparameter optimization for each ML model using the hyperopt-sklearn package [37], [38].

Step 4 (Performance Evaluation): To evaluate the performance of ML models under real-world conditions, we test detecting unknown rumors by separating training and validation datasets from the test datasets (i.e., they use different rumor events). To observe the performance difference, two types of tests are conducted: (1) cross validation using k-fold (e.g., $k = 10$) evaluation for known rumors, and (2) using other rumor event datasets for unknown rumor events. The rumor detection performances are discussed in the next section.

IV. SINGLE ML MODEL RUMOR DETECTION

This section evaluates the performance of ML models using 10-fold cross validation for detecting known rumor events, and training the models with other rumor event datasets for detecting unknown rumor events. In this section, we only used PHEME dataset, and use 80% of the dataset for training and 20% for testing. The results are shown in Figure 2. 2a and 2b show that knowing the rumor event can increase the accuracy and F1 score. Here, each ML model (total 13) is used for each rumor event (total 8) treating them as known, and then unknown. So in total, there are 104 results counted (i.e., the y-axis). Top-performing ML models are also shown in Tables 3 and 4 for known and unknown rumors,



(a) Overall F1 score

(b) Overall accuracy

(c) Events (known)

(d) Events (unknown)

(e) Models (known)

(f) Models (unknown)

(g) Features (known)

(h) Features (unknown)

FIGURE 2. Histogram measurement of rumor detection performances w.r.t. known vs. unknown detection, rumor events, ML models, and features. The x-axis is the F1-score values as bins and y-axis is count. (a) and (b) compare F1 score and accuracy of detecting known and unknown rumors, respectively. (c) and (d) compare F1 scores for detecting events as known and unknown rumors. (e) and (f) compare F1 scores for ML models to detect known and unknown rumors. (g) and (h) compare F1 scores to detect known and unknown rumors w.r.t. feature groups.

TABLE 4. Best ML model for detecting unknown rumors.

Rumors	Features			
	All	C	P	U
Crime				
Sydney Siege	LDA 0.87	**MLP 0.86**	RF 0.72	RF 0.73
Ottawa Shooting	GBM 0.85	**MLP 0.86**	RF 0.70	RF 0.74
Ferguson	LR 0.86	**LR 0.87**	GBM 0.73	RF 0.71
Charlie Hebdo	**GBM 0.86**	GBM 0.85	GBM 0.68	RF 0.72
Politic				
Putin Missing	MLP 0.89	**MLP 0.91**	RF 0.68	MLP 0.68
Entertainment				
Prince	RF 0.84	**RF 0.84**	RF 0.68	RF 0.72
Impact				
German Wings	MLP 0.85	**MLP 0.87**	RF 0.69	GBM 0.62
Mixed				
Ebola & Gurlitt	NB 0.83	**NB 0.85**	RF 0.70	AdaB 0.69

respectively (The best model and its F1 score are highlighted in bold font style). We further evaluate the performance of rumor detection models with respect to statistics, events, models, and features in this section.

A. BASIC STATISTICS

To better understand each of the rumor events, we show some key basic statistics in Table 5. In general, the non-rumor (NR) tweets have a higher favorite, retweet and friends counts (average 1.84, 1.09 and 1.12 times higher, respectively, without "Ebola & Gurlitt" event), as well as the number of keywords used and the number of unique keywords (labeled *Unique* in the table) across all tweets. Although we cannot make any conclusions based on these values, we can observe that non-rumors are more likely to be seen and spread by users in social media. However, looking at the individual event, the ratio differs significantly. For example, events "Prince" and "Ebola & Gurlitt" have a significantly lower ratio than other events.

B. EFFECTS OF RUMOR EVENTS AND TOPICS

Figures 2c and 2d compares the ML models detecting rumors when they are treated as known or unknown. Here, the y-axis represents the number of ML models in the F1-score ranges on the x-axis. From this, we observe that the order of best F1 scores for different events vary significantly, and it changes when we treat them as unknown rumors.

For detecting known rumors (Table 3), we observe that MLP achieved the highest F1 score for the "Charlie Hebdo" event, but LR was best for the "Ebola & Gurlitt" event, and GBM for "Ottawa Shooting" event. This indicates that even if the rumor events are from the same topic, such as "Crime" for the above examples, the best performing model varies. However, there is a group of models that appear more than others, which are NB, GBM, and RF for "Crime", RF for "Entertainment" and "Impact", and MLP for "Politic" and "Mixed" topics. Similarly, there is a group of models that appear more than others when detecting unknown rumors, which are MLP, LR, GBM for "Crime" topic, MLP for "Politic" and "Impact" topics, RF for "Entertainment" topic, and NB for "Mixed" topic. Although there are overlaps, the best model largely differs across the rumor topics.

TABLE 5. Statistics on rumor events.

Event	type	Favorite	Retweet	Friends	Keywords	Unique
Sydney Siege	NR	526	490	3820	2509	1855
	R	225	341	4912	1566	912
	Ratio	2.34	1.44	0.78	1.60	2.03
Ottawa Shooting	NR	243	353	4273	1580	1133
	R	105	325	2041	1286	839
	Ratio	2.32	1.09	2.09	1.23	1.35
Ferguson	NR	193	453	5176	3093	2498
	R	113	294	4513	1173	578
	Ratio	1.71	1.54	1.15	2.64	4.32
Charlie Hebdo	NR	233	455	4362	5183	4539
	R	103	355	1626	1347	703
	Ratio	2.26	1.28	2.68	3.85	6.46
Putin Missing	NR	9	16	1384	706	575
	R	6	15	4022	632	501
	Ratio	1.56	1.04	0.34	1.12	1.15
Prince	NR	3	2	746	24	15
	R	6	11	1598	794	785
	Ratio	0.53	0.21	0.47	0.03	0.02
German Wings	NR	109	238	2172	996	764
	R	50	226	6730	784	552
	Ratio	2.17	1.06	0.32	1.27	1.38
Ebola & Gurlitt	NR	1	2	1766	376	309
	R	93	262	5523	305	238
	Ratio	0.01	0.01	0.32	1.23	1.30

C. EFFECTS OF USING DIFFERENT ML MODELS

Figures 2e and 2f show the variations in performance to detect known and unknown rumors using different ML models, respectively. Similarly, we observe large variations of the best ML model to use (see the rank changes from Table 3 and 4.[6]) That is, it is not necessarily that the best performing model for known rumor detection would also be the best when used for unknown detection. Here, the y-axis count represents the number of rumor events.

D. EFFECTS OF RUMOR FEATURES

Figures 2g and 2h show performance variations with respect to different feature groups. Here, the y-axis represents the number of ML models and their performance against each rumor event, a total of 104 counts. Unlike other variables, we observe that either using all features or content-only feature group significantly improves the detection performance. Furthermore, using content-only features for some rumor events achieved a higher F1 score than using all features (i.e., in the cases of *Ferguson, German Wings, Putin Missing, and Sydney Siege* in Table 3). Interestingly, a similar observation was made for unknown rumor detection (i.e., even if we do not know what rumor we are looking for, the emerging rumor event may have similar characteristics). On the other hand,

[6]In Figure 2f, GBM is the best performer, but in Table 4, the best performer differs as it provides a better insight into individual rumor events.

using the user-based feature performed the worst. Looking at within each feature group, there is no one particular ML model that could be selected as the best performing one due to a large variation (as shown in Table 4). Hence, no one particular ML model is more sensitive to different feature groups than the others. In conclusion, our results show that regardless of detecting known or unknown rumors, using content-based features influenced the performance most significantly (unless the dataset used is skewed for known rumor detection, such as ''Prince'' event).

E. VERDICT

We compared the performance of rumor detection with respect to basic statistics, events and topics, different ML models, and rumor features. From those results, four main observations are made:

1) **Basic statistics cannot predict the performance of ML models**: We cannot estimate the performance of unknown rumor detection based on the R-to-NR ratio, which was somewhat possible for known rumor events.
2) **Degraded performance in general (Figures 2a and 2b)**: Unknown rumor detection performed poorly as shown by F1 scores achieved. For example with the ''German Wings'' event, the best F1 score is 0.867 using MLP as an unknown rumor, while the best F1 score is 0.968 using RF as a known rumor.
3) **The best performing ML model has changed (Figures 2c to 2f)**: The best performance achieved between known and unknown rumor detection has changed significantly. In fact, only the ''Politic'' topic achieved the best performance using the same model (i.e., MLP), and all other topics used a different ML model to achieve the best performance (see Table 3 and 4).
4) **Use content-based features or use all features (Figures 2g and 2h)**: Using content-based features and/or all features outperformed user and propagation features significantly in both known and unknown rumor detection.

Due to those limitations, a better approach is needed to improve the unknown rumor detection. To achieve this, we propose to use ES formed with well-performing ML models in the next section.

V. ENSEMBLE SOLUTION: MULTI-ML MODELS

To improve unknown rumor detection, we grasp the best properties of different ML models found in Section IV, and construct an ensemble solution (ES) by combining multi-ML models.

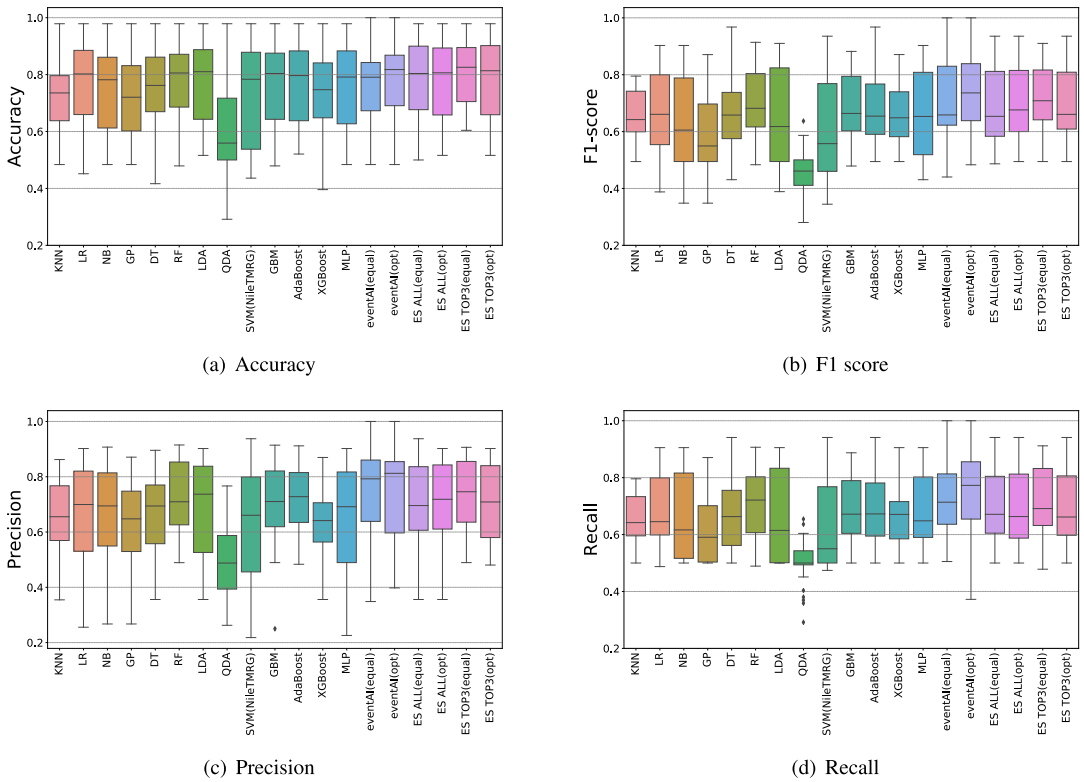
A. ENSEMBLE SOLUTION SETUP

There are many techniques to construct ensemble solutions, including weighting, voting, bagging, vogging, entropy, etc. [39], [40]. However, users must be aware of the limitations of each technique and the limitations of the way different classifiers can be used as part of the ensemble solution. In this paper, we implemented an ES by applying a weighted soft voting strategy. Because each classifier is making its own prediction rather than cooperating, a voting strategy is more feasible than other ensemble techniques. Further, soft voting allows the confidence of each classifier into account rather than making a binary decision as in hard voting. However, other ensemble techniques will be compared in our future work to determine the optimal rumor detection strategy using ESes. The process is captured in Algorithm 1, which shows the soft voting procedure used to make a decision in our developed ES. Later in the experiment (Section VI), we demonstrate that without a proper selection of classifiers and weight selection, not all ESes can detect the rumors the same. Here, function f describes our ES, where it is used to determine whether a given input dataset, D of size m, is either rumor or non-rumor. Also, a weight value W ($0 \leq W \leq 1$) is individually assigned to each ML model L_T, which represents the relative contribution of L_T because all models do not contribute equally (as shown in Section IV). Because the final result is not always 0 or 1, but in between, a threshold value T_{hold} is used to determine the agreed consensus of the votes (0.5 is used by default). This threshold value provides flexibility in adjusting the level of measured score to be in order to determine the rumor/non-rumor classification of the input data. $P(R|x_i)$ means the probability that the ith data from the dataset D is a rumor. This value can be obtained by calculating the prediction probability for each machine learning model. h_t is an array that stores the probability value of the ith data predicted by the algorithm L_i. z_t represents the final value accumulated by applying the weights W for each h_t of ith data.

Algorithm 1 ES Configuration Using Soft Voting

input : Dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots (x_m, y_m)\}$
input : Learning algorithms $L_1, \ldots L_T$
input : Weights for each L_i, $W_1, \ldots W_T$
input : Threshold T_{hold}
output: $f(D)$
for ($i = 1$; $i <= m$; i++) {
  for ($l = 1$; $l <= T$; l++) {
    $h_{ti} = L_t(D_i) = (P('\texttt{NR}'|X_i), P('\texttt{R}'|X_i))$
    $z_{ti} = \sum_{t=1}^{T} W_t h_{ti}$
    if $P('\texttt{R}'|X_i) \leq T_{hold}$ **then**
      $s(D_i) = \texttt{R}$
    else
      $s(D_i) = \texttt{NR}$
return $f(D) = \prod_{i=1}^{m} s(D_i)$

- **Model Selection** –
  - **All**: No prior knowledge of ML models and their rumor detection performance.
  - **Top** N: Has each ML model performance measured and be able to rank them.



(a) Accuracy

(b) F1 score

(c) Precision

(d) Recall

FIGURE 3. **Proposed ES vs. others (known rumors).**

- **Weight Values** –
  - **Equal**: User does not have access to evaluate and configure ML models for rumor detection.
  - **Optimized**: User can pre-train and evaluate ML models using different rumor events.

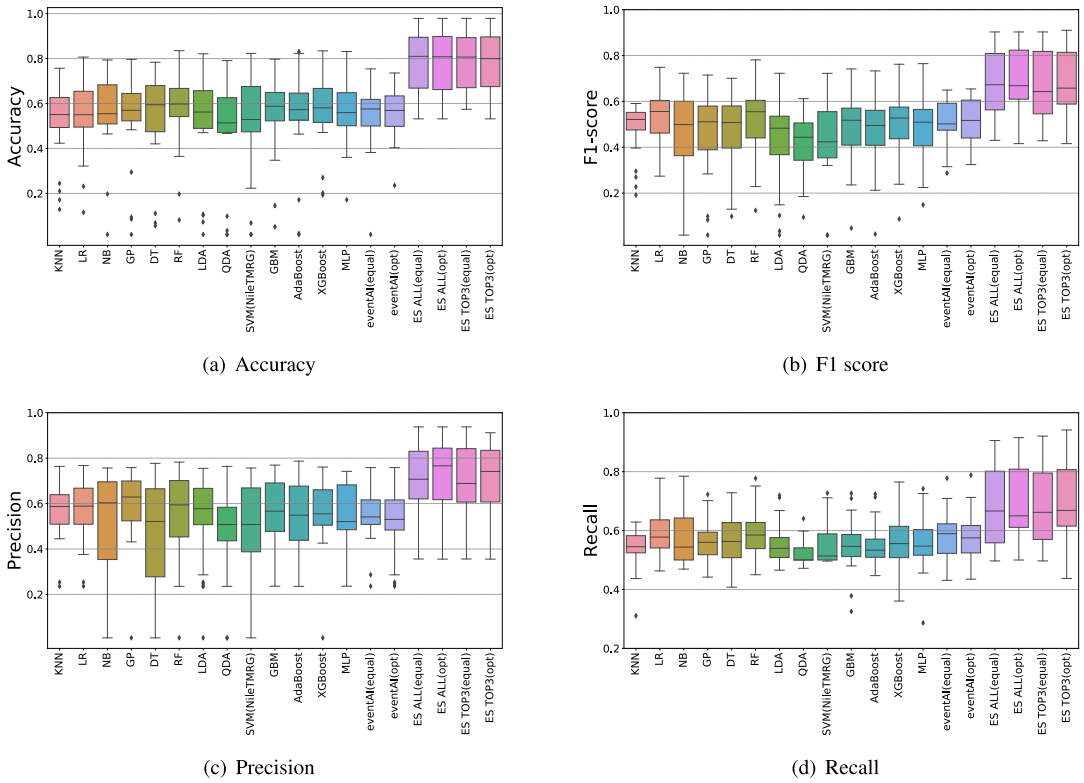
Top N: We can select the top N models to form the ES, because some models perform poorly for all rumor events and features, as seen in Section IV (i.e., there is always another model that performs better). Hence, incorporating those under-performing ones may not necessarily improve rumor detection, and possibly negatively impact the performance. The top N models are selected based on the performance observed in the past (i.e., we must evaluate their performance prior to assembling them into the ES). As such, this approach will add the overhead for pre-evaluating the ML models for selection.

Weight Optimization: The weight values can be optimized to improve the performance as demonstrated in [41], [42]. In this paper, this is achieved through hyper-parameter optimization using 25% of the dataset (as mentioned in Section III). Because some models may perform better than others (as shown in Section IV, and further demonstrated later in Section VI), they can be given higher weight values proportional to their relative rumor detection performance. Based on the results shown in Section IV, their F1 score performance used to determine the weight value ratio (e.g., if we use model A and B with F1 cores 0.8 and 0.6 respectively, then their weights are assigned 0.8:0.6 ratio in the ES), suggesting a linear relationship between the weights and F1 scores. Note that, this will require further training using the existing dataset,[7] and also if the training set is biased, the end result may not enhance the performance. Hence, different weight assignment strategies should be explored in future work. But as we observe later in Section VI, using a large-sized dataset such as PHEME can lead to reasonably good performance.

VI. EXPERIMENTAL ANALYSIS

In our experiment, we compare our proposed ES strategies against both classical methods (i.e., single ML model-based solutions) and recent ES solutions (e.g., evenAI) using real data sets (i.e., PHEME and RE2019): (1) the proposed ES strategies consist of ALL(equal), ALL(opt), TOP3(equal) and TOP3(opt) as described in Section V-A, (2) single ML models including NileTMRG as described in Section III, and (3) eventAI (a recent ES implementation consisting of LR, RF and SVM, achieving the best performance as shown in [7]). To provide a fair comparison, all models used base ML models without any fine-tuning, which is true for our implemented ESes as well. Next, we examine their performance when we use a smaller RE2019 dataset in Section VI-B. Finally, we compare different ES constructions (our proposed

[7] A ML model may be trained and optimized already, but if there is no available dataset, we cannot optimize weights.



(a) Accuracy (b) F1 score

(c) Precision (d) Recall

FIGURE 4. Proposed ES vs. others (unknown rumors).

strategies versus random and worst classifier selections) in Section VI-C.

A. EXPERIMENT WITH PHEME DATASET

In this experiment, we set training, validation and testing dataset ratio at 60%, 20% and 20%, respectively. The results are shown in Figures 3 and 4 for known and unknown rumor detection, respectively.

Known rumor detection: When the rumor is known, the ES (regardless of the strategies) has no significant difference as shown in Figure 3. This is also the same for eventAI.

Unknown rumor detection: When detecting unknown rumors, however, the proposed ES strategies performed significantly better than others including eventAI as shown in Figure 4. Another observation is that the performances across different strategies of the ES are relatively the same. This indicates that if we have a large dataset available, such as PHEME, it seems feasible to combine already trained ML models into an ES without requiring further tuning for weights or ranking them. However, selecting a few, such as eventAI, the user has to be careful to choose the best-performing ones such as selecting optimally performing ones. As shown in Table 4, we anticipate the ES performs better than single ML models as different models in the ES pick up different characteristics of rumors to give a combined assessment, improving the overall classification.

1) STATISTICAL ANALYSIS

The Kruskal-Wallis (KW) test was used to conduct statistical analysis. The KW test is a non-parametric method that responds to the one-way analysis of variance (ANOVA). The KW test allows us to compare the distributions of multiple groups (i.e., ML models and the ES) on a dependent variable that is measured (i.e., rumor detection performance). Hence, this test validates that results observed in Figure 4 are statistically different (i.e., the proposed ES is better than the others). To adjust p-values in multiple comparison procedures, we used the Bonferroni correction method [43].

First, our proposed ES is compared with single ML models as shown in Figure 5.[8] This test confirms our results shown in Figure 4 such that the performance of our proposed ES is significantly different (better) than **ALL** single ML models in terms of accuracy, and a similar result was also observed for F1 scores as well.

Second, we compare our proposed ES against eventAI as shown in Figure 6. It also clearly shows that our proposed ESes, regardless of the strategies used, are better performers than eventAI. As the eventAI solution was used with a

[8]NileTMRG is presented as SVM, it's underlying model. Also, the propagation and user feature groups are omitted, as well as precision and recall data, as not many differences are observed. Full details can be retrieved from our technical paper from https://github.com/hksecurity/RumorDetection-with-Multi-ML-Models.

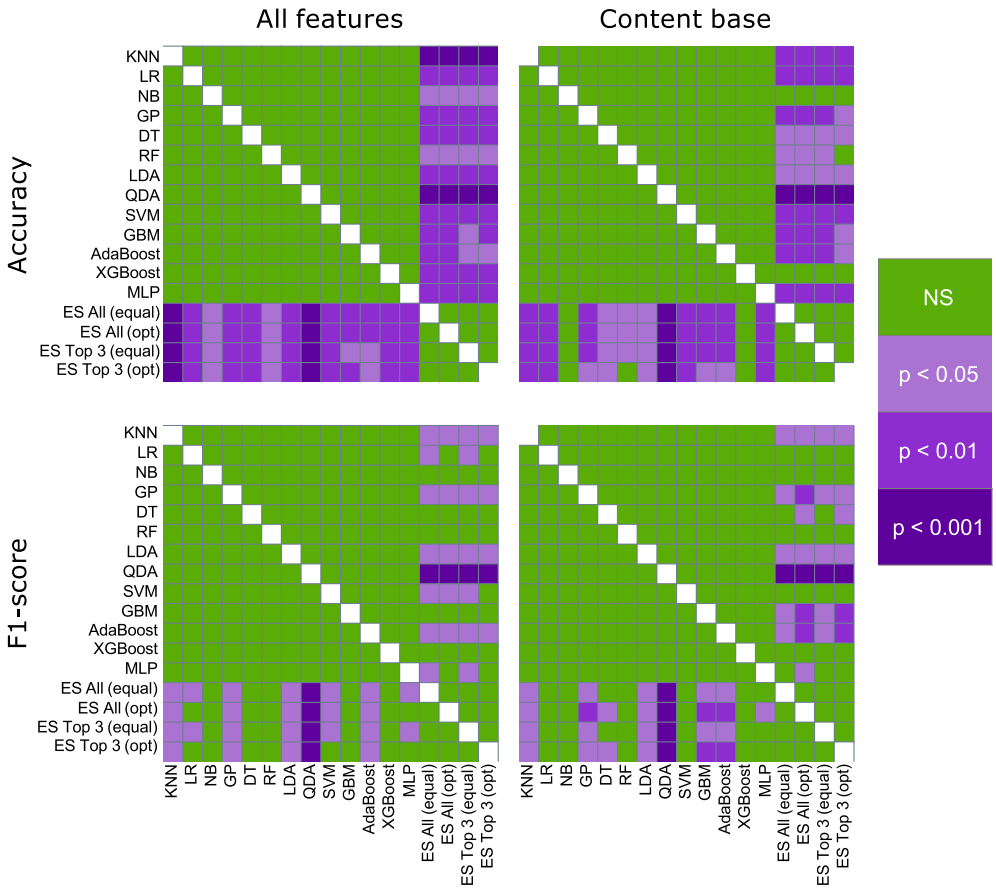


FIGURE 5. KW test: proposed ES vs. Single ML models.

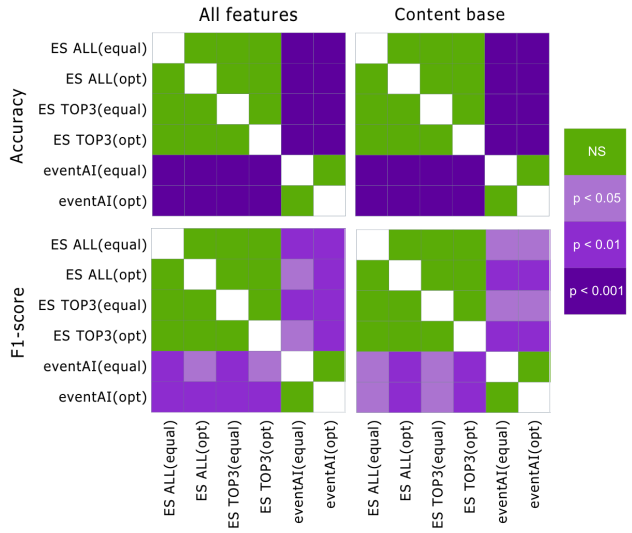


FIGURE 6. KW test: proposed ES vs. eventAI.

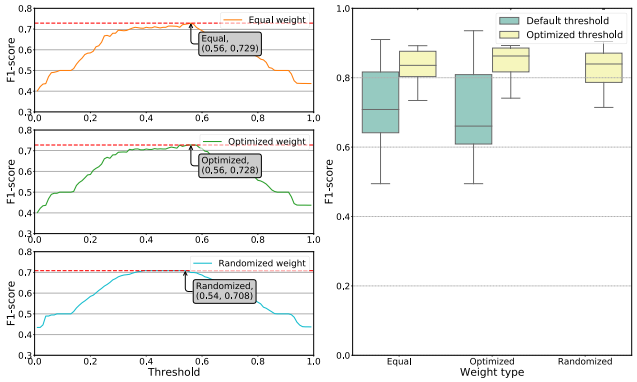
different set of data, the performance has changed significantly, showing that not all ensemble solutions can improve the unknown rumor detection performance.

2) SOFT VOTING CONFIGURATION

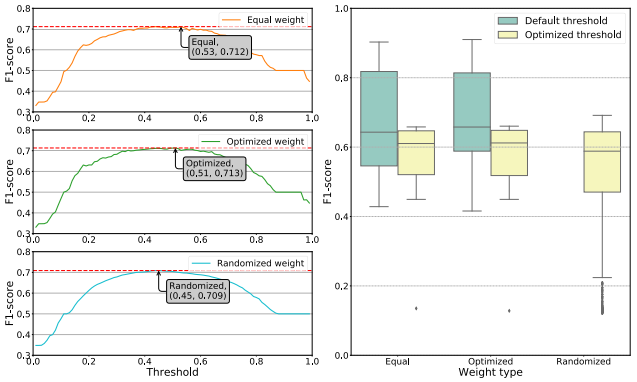
In addition to the proposed ES strategies, performance may vary depending on the way we determine the outcomes using our selected soft voting approach (i.e., the impact of threshold or weight values). To evaluate this, we performed the following experiment setups using the Top3 ML models.

- **Optimized threshold** –
  - ***the global optimized threshold*** means the optimized threshold for all experiments, and ***the local optimized threshold*** means the optimized threshold for each event experiment.
  - Based on our findings from the global /local optimized threshold calculations above (i.e., the best performance through validation set by increasing the threshold value from 0 to 1 by 0.01), we experiment using the global/local optimized threshold.
- **Randomized weights** –
  - We experimented with 100 random weight values.

The results are shown in Figures 7 and 8 for global and local threshold values, respectively. As a result of the experiment, optimized thresholds performed better in the known event test than in the unknown event test. Also, both global and local optimized thresholds performed better than the

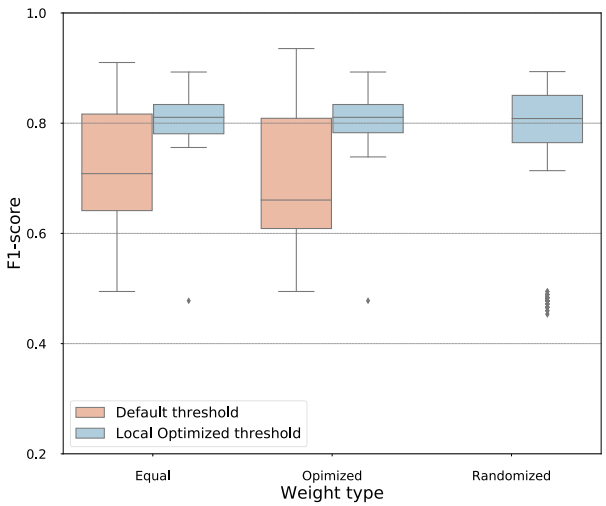


(a) Global threshold in known events

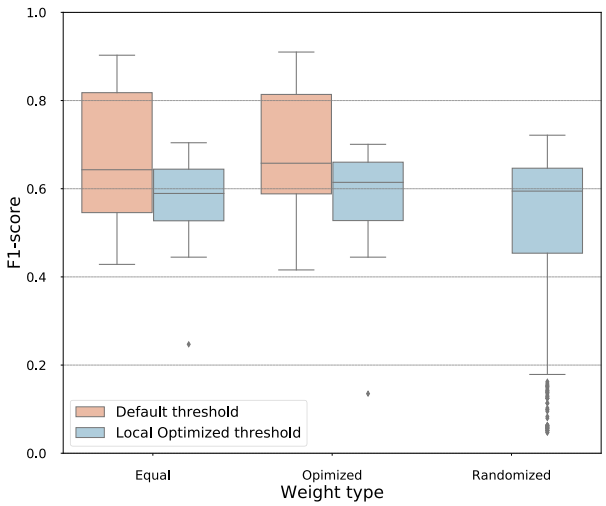


(b) Global threshold in unknown events

FIGURE 7. Comparing F1-scores for global thresholds when detecting rumors.



(a) Local threshold in known events



(b) Local threshold in unknown events

FIGURE 8. Comparing F1-scores for local thresholds when detecting rumors.

default thresholds in the known event test. And the optimized weights performed similarly or better than equal weights or randomized weights. These results indicate that it is important to ensure that soft voting weights are optimized to improve rumor detection.

B. EXPERIMENT WITH RE2019 DATASET

In this experiment, we trained our ES using the PHEME dataset, removed any overlapping events between the PHEME and RE2019 Twitter datasets and tested for unknown rumor detection. The parameters for the setup remained the same as before. The results are shown in Figures 9 and 10 for RE2019 dataset using Twitter and Reddit posts, respectively. Due to the small-sized dataset, we could not conclude any statistical difference (i.e., the variance is too large).

1) TWITTER FROM RE2019

Our result shows that our proposed ES still performs better than single ML models, as well as eventAI. Further, improvements are only significant using ES TOP3 strategies (for both *opt* or *equal* weighting). We suspect that since the dataset size is small, some models perform very poorly and therefore reduces the performance when all models are used.

2) REDDIT FROM RE2019

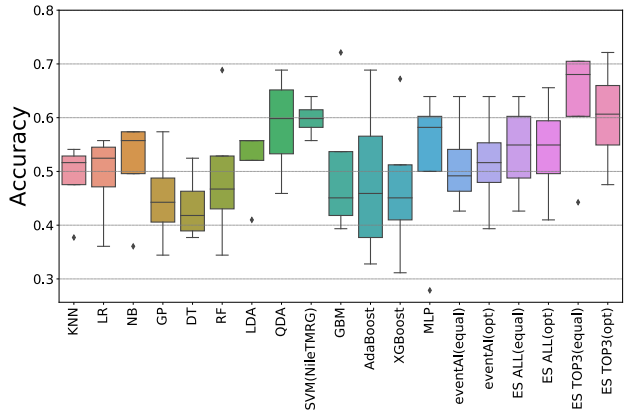
To evaluate the Reddit data from RE2019, we first extracted features from each Reddit post. We mapped the following features from Reddit posts:

- **Title**: has_question_mark, has_exclamation_mark, unigram_bow_vector, and pos_tagging.
- **URL**: has_URL.
- **Subreddit_subscribers**: followers_count.
- **Num_comments**: statuses_count.

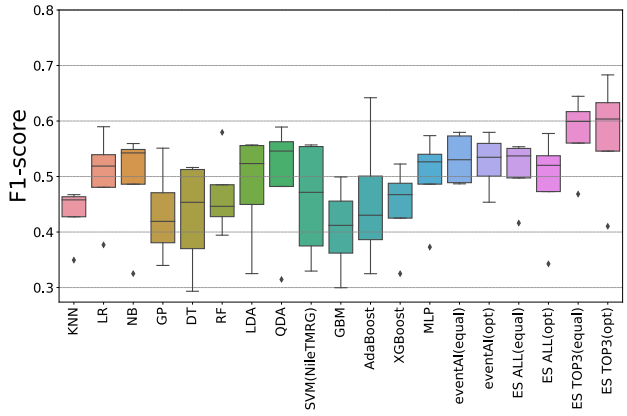
As we can see, not all features used for Twitter are present. However, better performance was still observed using ES TOP3 in comparison to single ML models and eventAI, as shown in Figure 10. We suspect that the top three best performing ML models when used for ES can compliment and identify rumors that were otherwise would have been undetected.

C. COMPARING ES CONFIGURATIONS

We examine the performance of ESes when their configurations vary. In our previous experiments, we used optimized

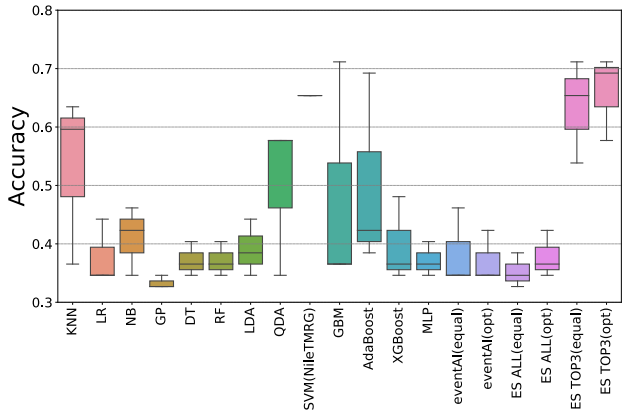


(a) Accuracy

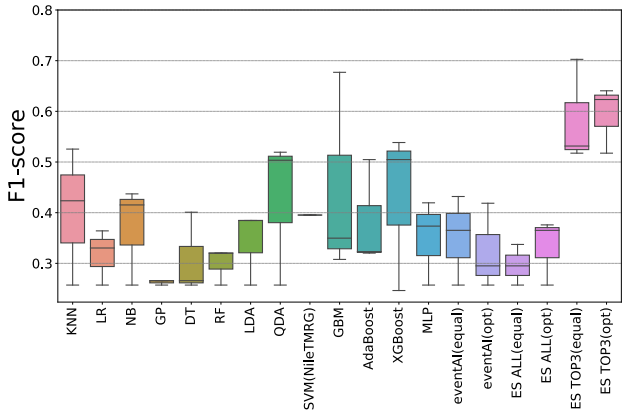


(b) F1 score

FIGURE 9. Proposed ES vs. others (RE2019 Twitter).



(a) Accuracy



(b) F1 score

FIGURE 10. Proposed ES vs. others (RE2019 Reddit).

values for the best results. However, users may believe that regardless of the ES configuration, using any combinations of the ML models (classifiers) as an ensemble could still be better at detecting unknown rumors. So, we carried out ES configuration comparisons as shown in Figure 11 using both PHEME and RE2019 datasets together. We setup three ES configurations: (1) Worst, (2) Random, and (3) Best. *Worst* refers to the collection of worst-performing ML models to construct the ES, *Random* refers to randomly selected ML models to construct the ES, and *Best* refers to our previously described approach (as shown in Section V). For the threshold values, we have two modes of the same (i.e., equal weights) or optimized (as described in Section V). We vary the number of combined models from two to six, because if the number of models increases, the computational overhead (as well as configurations) increases. Therefore, to minimize the computation overhead, we assume that using the minimal number of ML models for the ES is the user's goal.

Figures 11a and 11b shows their comparisons in the context of detecting known rumors. Since detecting known rumors has similar results, *Random* and *Best* perform similarly. However, if the user somehow selected the worst-performing models, then both accuracy and F1 score are significantly reduced. Regardless, using the optimized ES outperforms both worst and randomly configured ESes. We also notice that as the number of models used increases (along the x-axis), their rumor detection performance becomes similar. This is the same observation we made in our previous experiment when using all models performed relatively better than any of the single models. This also minimizes the noise created by not so well performing models as well.

Figures 11c and 11d shows their comparisons in the context of detecting unknown rumors. Similarly, with detecting known rumors, *Worst* and *Random* configurations had poor detection performance. This indicates that depending on how the ES is configured (i.e., the selection of ML models and the threshold value), the rumor detection particularly unknown ones can be affected severely.

VII. DISCUSSION

A. RUMOR DATASET

Gathering the dataset for rumors is a challenging task due to the lack of automated processes to annotate them. Because existing rumor detectors are not reliable to detect unknown rumor events, they cannot be used to annotate rumors. Although the proposed ES is able to improve the detection

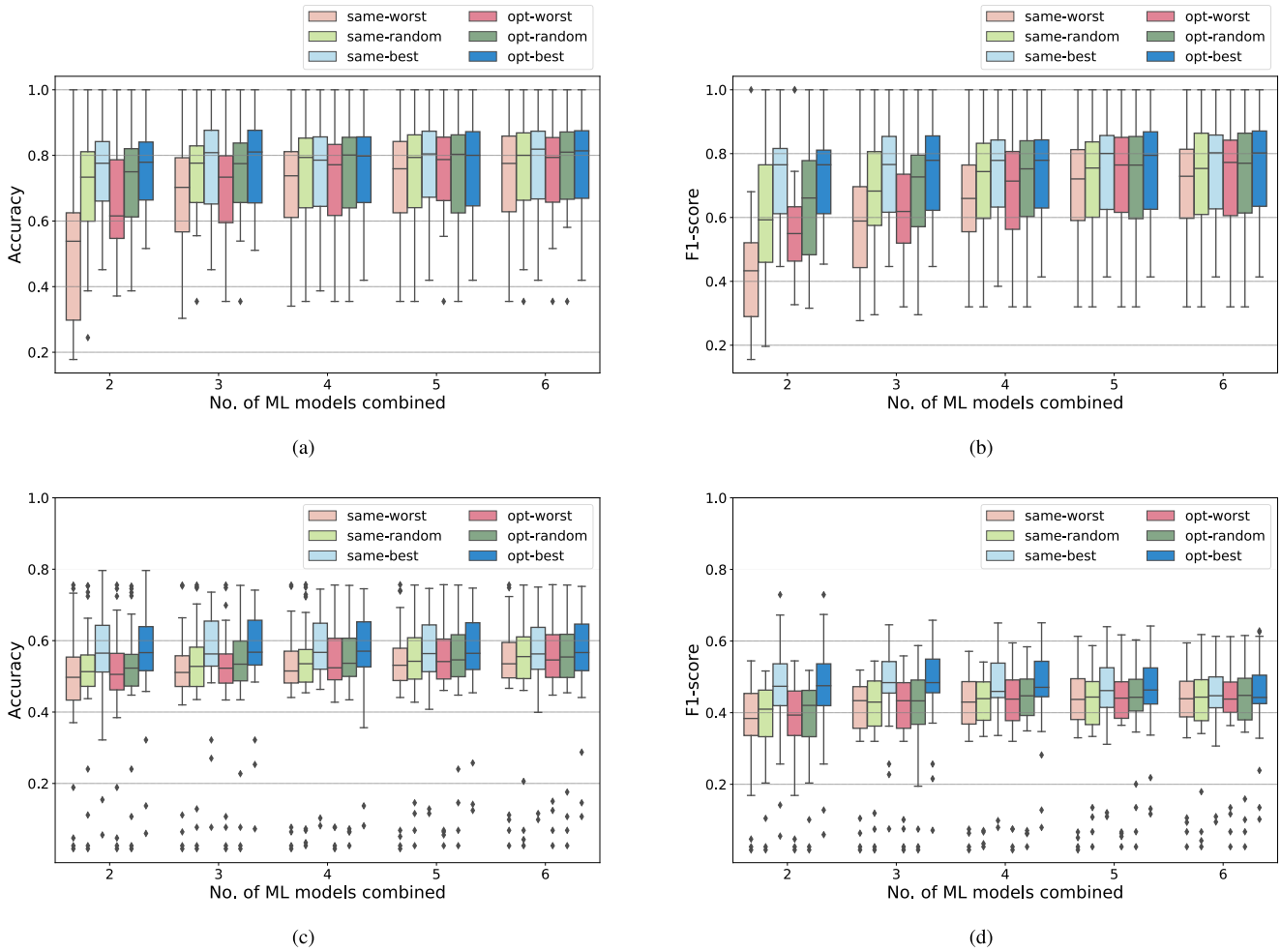


FIGURE 11. Comparing ESes (worst, random and optimized). (a) and (b) compare accuracy and F1-scores for known rumor detection, respectively. (c) and (d) compare accuracy and F1-scores for unknown rumor detection, respectively.

of rumors for unknown rumor events, it still requires training using known rumor data. To address this problem, techniques such as rumor generation using GAN [32] can be used to enhance the classification of unknown rumor events further. This limitation affects the use of certain models and techniques, as well. For example, neural network algorithms (e.g., RNN, LSTM, GRU, etc.) require parsing the rumor dataset into a sequence, which is not presented in some datasets. Hence, rumor generation and collection are still challenging tasks, which will be investigated in our future work.

Although we used some correlated datasets based on rumor topics (as shown in Table 1), rumor detection performance varied significantly across the events we examined, as shown in Table 4. This implies that rumor events themselves do not necessarily share features that are correlated within the same rumor topic. Hence, other correlation amongst rumor topics needs to be investigated.

B. RUMOR FEATURE

Our experimental results showed that content-based features enhanced the performance of all models significantly compared to other features. However, in many cases combining all features did not necessarily improve the performance over the content-based feature (they are not statistically different in performance). We suspect that because using other features, P and U, did not improve the detection performance, adding those features to a classifier may introduce noise in detection rather than improving the performance. This indicates the importance of understanding the context of the rumor. To further enhance the detection performance, the content-based feature can be enriched by incorporating contextual information (e.g., context-aware rumor detection using sentimental information). Zubiaga *et al*. [6] incorporated natural language processing (NLP) to predict the veracity of the rumor, which performed better when compared to without using the NLP. Similarly, we can enhance content-based features by incorporating context-aware analysis of the rumor.

Propagation and user-based features were not sufficiently effective in rumor detection in our experiments. As shown in Table 5, rumor events had quite different base metrics that contribute towards the propagation and user-based features. However, the rumor detection performance by ML models was not affected by them. It could be due to rumor spread having similar properties as non-rumor spread (for propagation).

Although rumor spreading users had much less metric values (e.g., fewer friends, retweets, and favorite counts), the overall characteristics may not be significant for the tested models to identify them. A further look into the usefulness of features will be examined in our future work.

C. BEST PERFORMING ML MODELS

Using the ES (regardless of any strategies to combine them) is the suggested solution when detecting unknown rumors, as it outperforms all other single model-based detectors as shown in Section V. This clearly demonstrates that using the ES overcomes the limitations of using single ML models only, which exhibits biased detection for a subset of rumor features when used, reducing the performance when used against unknown rumors. If limited features are given (e.g., User only, or Propagation only), then many of the tested models will perform relatively the same. If only user-based features are given, a few models to avoid using are DT, LDA, QDA, and MLP. If only propagation-based features are given, then NB, LDA, and SVM are not recommended.

Although we did not find any strong relationship between rumor topics and single ML models in rumor detection performance, the following models could be suggested based on our statistical test shown in Figure 5: NB, RF, GBM, and XGBoost. Although those model's accuracy is significantly less than using the ES, their F1 score achieved relatively the same with the ES. Other models not listed above are not recommended as their performance is significantly less than the ES.

D. CONFIGURING THE ENSEMBLE SOLUTION

In this paper, we discussed the four ES strategies (ALL(equal), ALL(opt), TOP3(equal) and TOP3(opt)), but found that using any of them generated similar performance. Hence, the weight assignment and choosing N best ML models do not necessarily improve the performance. That is, even if N is a small number, it can be helpful to reduce training and testing times. However, we assumed that the selected models are already optimized to detect rumors. If we are given off-the-shelf ML models only, then hyper-parameter configuration could impact the performance significantly. Moreover, because the detection performance varies when rumor topics change, it may also be possible to dynamically adjust the weights between ML models in the ES based on empirical studies. To investigate further to formulate the most effective ES, we will investigate the use of other ML models with and without optimization, as well as in conjunction with other rumor topic-related features in our future work.

VIII. CONCLUSION

Many different ML models have been proposed to detect rumors. But using those models to detect new and emerging rumor topics is still a challenging task today. We have validated this in our experimental analysis, which showed that there is no one ML model that can outperform others, especially when various rumor topics are considered. We also discovered that using content-based features only might be a better strategy than using all features for using an ML model to detect unseen rumors specifically.

To address the aforementioned problems, we proposed ensemble solutions (ESes) combining multi-ML models together to work cooperatively to detect rumors of various topics with different features given. The experimental results showed that there is a significant improvement in the performance when using the ES to detect rumors that are previously unknown, achieving up to 0.79 average F1 score, compared to 0.58 using a single ML model (i.e., an increase of 0.21 average F1 score). With the enhanced rumor detection performance, the ES would be suitable and practical for early rumor detection without needing any of the unknown rumor data, which can take a significant time to identify and collect, while the caused damage increases.

REFERENCES

[1] S. Kumar, R. West, and J. Leskovec, "Disinformation on the Web: Impact, characteristics, and detection of wikipedia hoaxes," in *Proc. 25th Int. Conf. World Wide Web (WWW)*, 2016, pp. 591–602.

[2] H. Webb, P. Burnap, R. Procter, O. Rana, B. C. Stahl, M. Williams, W. Housley, A. Edwards, and M. Jirotka, "Digital wildfires: Propagation, verification, regulation, and responsible innovation," *ACM Trans. Inf. Syst.*, vol. 34, no. 3, pp. 15:1–15:23, 2016.

[3] A. Zubiaga, M. Liakata, R. Procter, G. W. S. Hoi, and P. Tolmie, "Analysing how people orient to and spread rumours in social media by looking at conversational threads," *PLoS ONE*, vol. 11, no. 3, Mar. 2016, Art. no. e0150989.

[4] N. Grinberg, K. Joseph, L. Friedland, B. Swire-Thompson, and D. Lazer, "Fake news on Twitter during the 2016 U.S. presidential election," *Science*, vol. 363, no. 6425, pp. 374–378, Jan. 2019.

[5] S. Wang, Q. Kong, Y. Wang, and L. Wang, "Enhancing rumor detection in social media using dynamic propagation structures," in *Proc. IEEE Int. Conf. Intell. Secur. Informat. (ISI)*, Jul. 2019, pp. 41–46.

[6] A. Zubiaga, M. Liakata, and R. Procter, "Exploiting context for rumour detection in social media," in *Proc. Int. Conf. Social Informat. (SocInfo)*. Cham, Switzerland: Springer, 2017, pp. 109–123.

[7] G. Gorrell, E. Kochkina, M. Liakata, A. Aker, A. Zubiaga, K. Bontcheva, and L. Derczynski, "SemEval-2019 task 7: RumourEval, determining rumour veracity and support for rumours," in *Proc. 13th Int. Workshop Semantic Eval. (RumourEval)*, Jun. 2019, pp. 845–854.

[8] C. Buntain and J. Golbeck, "Automatically identifying fake news in popular Twitter threads," in *Proc. IEEE Int. Conf. Smart Cloud (SmartCloud)*, Nov. 2017, pp. 208–215.

[9] C. Castillo, M. Mendoza, and B. Poblete, "Information credibility on Twitter," in *Proc. 20th Int. Conf. World Wide Web (WWW)*, 2011, pp. 675–684.

[10] S. Liu, Y. Wang, J. Zhang, C. Chen, and Y. Xiang, "Addressing the class imbalance problem in Twitter spam detection using ensemble learning," *Comput. Secur.*, vol. 69, pp. 35–49, Aug. 2017.

[11] T. N. Nguyen, C. Li, and C. Niederée, "On early-stage debunking rumors on Twitter: Leveraging the wisdom of weak learners," in *Social Informatics*. Cham, Switzerland: Springer, 2017, pp. 141–158.

[12] A. Zubiaga, A. Aker, K. Bontcheva, M. Liakata, and R. Procter, "Detection and resolution of rumours in social media: A survey," *ACM Comput. Surv.*, vol. 51, no. 2, pp. 32:1–32:36, Jun. 2018.

[13] A. Kumar and S. R. Sangwan, "Rumor detection using machine learning techniques on social media," in *Proc. Int. Conf. Innov. Comput. Commun. (ICICC)*. Singapore: Springer, 2019, pp. 213–221.

[14] Z. Yang, C. Wang, F. Zhang, Y. Zhang, and H. Zhang, "Emerging rumor identification for social media with hot topic detection," in *Proc. 12th Web Inf. Syst. Appl. Conf. (WISA)*, Sep. 2015, pp. 53–58.

[15] Z. Zhao, P. Resnick, and Q. Mei, "Enquiring minds: Early detection of rumors in social media from enquiry posts," in *Proc. 24th Int. Conf. World Wide Web (WWW)*, 2015, pp. 1395–1405.

[16] L. Tian, X. Zhang, Y. Wang, and H. Liu, "Early detection of rumours on Twitter via stance transfer learning," in *Advances in Information Retrieval*, J. M. Jose, E. Yilmaz, J. Magalhães, P. Castells, N. Ferro, M. J. Silva, and F. Martins, Eds. Cham, Switzerland: Springer, 2020, pp. 575–588.

[17] E. Ferrara, O. Varol, C. Davis, F. Menczer, and A. Flammini, "The rise of social bots," *Commun. ACM*, vol. 59, no. 7, pp. 96–104, Jul. 2016.
[18] T. G. Dietterich, "Ensemble methods in machine learning," in *Multiple Classifier Systems*. Berlin, Germany: Springer, 2000, pp. 1–15.
[19] E. Bauer and R. Kohavi, "An empirical comparison of voting classification algorithms: Bagging, boosting, and variants," *Mach. Learn.*, vol. 36, nos. 1–2, pp. 105–139, 1999.
[20] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *J. Artif. Intell. Res.*, vol. 11, pp. 169–198, Aug. 1999.
[21] L. Xu, A. Krzyzak, and C. Y. Suen, "Methods of combining multiple classifiers and their applications to handwriting recognition," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, no. 3, pp. 418–435, May/Jun. 1992.
[22] Y. Geng, Z. Lin, P. Fu, and W. Wang, "Rumor detection on social media: A multi-view model using self-attention mechanism," in *Proc. Int. Conf. Comput. Sci. (ICCS)*. Cham, Switzerland: Springer, 2019, pp. 339–352.
[23] S. Kwon, M. Cha, K. Jung, W. Chen, and Y. Wang, "Prominent features of rumor propagation in online social media," in *Proc. IEEE 13th Int. Conf. Data Mining*, Dec. 2013, pp. 1103–1108.
[24] Y. Yang, K. Niu, and Z. He, "Exploiting the topology property of social network for rumor detection," in *Proc. 12th Int. Joint Conf. Comput. Sci. Softw. Eng. (JCSSE)*, Jul. 2015, pp. 41–46.
[25] G. Giasemidis, C. Singleton, I. Agrafiotis, J. R. Nurse, A. Pilgrim, C. Willis, and D. V. Greetham, "Determining the veracity of rumours on Twitter," in *Proc. Int. Conf. Social Informat. (SocInfo)*. Cham, Switzerland: Springer, 2016, pp. 185–205.
[26] Y. Liu, X. Jin, H. Shen, and X. Cheng, "Do rumors diffuse differently from non-rumors? A systematically empirical analysis in Sina Weibo for rumor identification," in *Proc. Pacific-Asia Conf. Knowl. Discovery Data Mining (PAKDD)*. Cham, Switzerland: Springer, 2017, pp. 407–420.
[27] S. Kwon, M. Cha, and K. Jung, "Rumor detection over varying time windows," *PLoS ONE*, vol. 12, no. 1, Jan. 2017, Art. no. e0168344.
[28] J. Ma, W. Gao, Z. Wei, Y. Lu, and K.-F. Wong, "Detect rumors using time series of social context information on microblogging websites," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2015, pp. 1751–1754.
[29] H. Guo, J. Cao, Y. Zhang, J. Guo, and J. Li, "Rumor detection with hierarchical social attention network," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, Oct. 2018, pp. 943–951.
[30] T. Chen, X. Li, H. Yin, and J. Zhang, "Call attention to rumors: Deep attention based recurrent neural networks for early rumor detection," in *Proc. Pacific-Asia Conf. Knowl. Discovery Data Mining (PAKDD)*. Cham, Switzerland: Springer, 2018, pp. 40–52.
[31] N. Ruchansky, S. Seo, and Y. Liu, "CSI: A hybrid deep model for fake news detection," in *Proc. ACM Conf. Inf. Knowl. Manage. (CIKM)*, Nov. 2017, pp. 797–806.
[32] J. Ma, W. Gao, and K.-F. Wong, "Detect rumors on Twitter by promoting information campaigns with generative adversarial learning," in *Proc. 28th Int. Conf. World Wide Web Conf. (WWW)*, 2019, pp. 3049–3055.
[33] T. Chen, H. Chen, and X. Li, "Rumor detection via recurrent neural networks: A case study on adaptivity with varied data compositions," in *Proc. Pacific-Asia Conf. Knowl. Discovery Data Mining (PAKDD)*. Cham, Switzerland: Springer, 2018, pp. 121–127.
[34] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," *Emerg. Artif. Intell. Appl. Comput. Eng.*, vol. 160, no. 1, pp. 3–24, 2007.
[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, and D. Cournapeau, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Nov. 2011.
[36] M. Kay, S. N. Patel, and J. A. Kientz, "How good is 85%?: A survey tool to connect classifier evaluation to acceptability of accuracy," in *Proc. 33rd Annu. ACM Conf. Hum. Factors Comput. Syst. (CHI)*, 2015, pp. 347–356.
[37] J. Bergstra, B. Komer, C. Eliasmith, D. Yamins, and D. D. Cox, "Hyperopt: A Python library for model selection and hyperparameter optimization," *Comput. Sci. Discovery*, vol. 8, no. 1, Jul. 2015, Art. no. 014008.
[38] B. Komer, J. Bergstra, and C. Eliasmith, "Hyperopt-sklearn," in *Automated Machine Learning*. Cham, Switzerland: Springer, 2019, pp. 97–111.
[39] L. Rokach, *Pattern Classification Using Ensemble Methods*, vol. 75. Singapore: World Scientific, 2010.
[40] H.-J. Li, Z. Bu, Z. Wang, J. Cao, and Y. Shi, "Enhance the performance of network computation by a tunable weighting strategy," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 3, pp. 214–223, Jun. 2018.
[41] J. Heinermann and O. Kramer, "Machine learning ensembles for wind power prediction," *Renew. Energy*, vol. 89, pp. 671–679, Apr. 2016.
[42] H.-J. Li, Z. Bu, Z. Wang, and J. Cao, "Dynamical clustering in electronic commerce systems via optimization and leadership expansion," *IEEE Trans. Ind. Informat.*, vol. 16, no. 8, pp. 5327–5334, Aug. 2020.
[43] M. A. Napierala, "What is the Bonferroni correction," *AAOS Now*, vol. 6, no. 4, p. 40, 2012.

YOUNGHWAN KIM received the B.S. degree from the Department of Mechanical Engineering, Naval Academy, and the M.S. degree from the Department of Computer Science, Korea National Defense University. He is currently pursuing the Ph.D. degree with the School of Cybersecurity, Korea University, under the supervision of H. K. Kim. His research interests include data mining, rumor detection, and machine learning.

HUY KANG KIM (Member, IEEE) received the B.S. degree in industrial management, the M.S. degree in industrial engineering, and the Ph.D. degree in industrial and systems engineering from the Korea Advanced Institute of Science and Technology (KAIST), in 1998, 2000, and 2009, respectively. He founded A3 Security Consulting, the first information security consulting company in South Korea, in 1999. He is currently a Professor with the School of Cybersecurity, Korea University. Before joining Korea University, he was a Technical Director (TD) and the Head of Information Security Department of NCSOFT, from 2004 to 2010, one of the most famous MMORPG companies in the world. His recent research is focused on solving many security problems in online games based on the user behavior analysis.

HYOUNGSHICK KIM (Associate Member, IEEE) received the B.S. degree from the Department of Information Engineering, Sungkyunkwan University, in 1999, the M.S. degree from the Department of Computer Science, KAIST, in 2001, and the Ph.D. degree from the Computer Laboratory, University of Cambridge, in 2012. After completing his Ph.D., he worked as a Postdoctoral Fellow with the Department of Electrical and Computer Engineering, The University of British Columbia. He previously worked for Samsung Electronics Company Ltd., as a Senior Engineer, from 2004 to 2008. He is currently an Associate Professor with the Department of Computer Science and Engineering, College of Software, Sungkyunkwan University. He is also working as a Distinguished Visiting Researcher with CSIRO Data61. His current research interests include usable security, blockchain, and software security.

JIN B. HONG (Member, IEEE) received the Ph.D. degree in computer science from the University of Canterbury, New Zealand. He is currently a Lecturer with the Department of Computer Science and Software Engineering, The University of Western Australia, Australia. His research interests include security modeling and analysis of computer and networks, including cloud computing, SDN and IoT, and moving target defense.