Ensemble Learning for Detecting Fake Reviews

Luis Gutierrez-Espinoza[1], Faranak Abri[1], Akbar Siami Namin[1], Keith S. Jones[2], and David R. W. Sears[3]
[1]Department of Computer Science, [2]Department of Psychological Sciences, [3]Performing Arts Research Lab
Texas Tech University
{*Luis.Gutierrez-Espinoza, faranak.abri, akbar.namin, keith.s.jones, david.sears*}@*ttu.edu*

***Abstract*—Customers represent their satisfactions of consuming products by sharing their experiences through the utilization of online reviews. Several machine learning-based approaches can automatically detect deceptive and fake reviews. Recently, there have been studies reporting the performance of ensemble learning-based approaches in comparison to conventional machine learning techniques. Motivated by the recent trends in ensemble learning, this paper evaluates the performance of ensemble learning-based approaches to identify bogus online information. The application of a number of ensemble learning-based approaches to a collection of fake restaurant reviews that we developed show that these ensemble learning-based approaches detect deceptive information better than conventional machine learning algorithms.**

***Index Terms*—Ensemble learning, deception detection.**

I. Introduction

There are several factors that contribute to the success or failure of a business, and customer satisfaction is always listed as one of the most important ones. Customer satisfactions and good reputation is a matter of life and death for many businesses. It is important to take customer's reviews into account seriously and address their concerns concisely. However, online reviews can also be abused by adversaries for different reasons. It is therefore important to distinguish between genuine and dubious reviews. The automatic detection of fake online reviews is an inherent instance of a simple binary classification problem. As a result, conventional machine learning-based classification or simple statistical-based approaches can be applicable to this problem. Given the natural language-based nature of reviews and the hidden features that might be latent for explicit modeling, the accuracy might vary.

The use of the most accurate classification algorithms available for deception is essential in order to design the best possible detection platforms. Conventional machine learning algorithms have been extensively used in the literature concerned with the detection of deceptive and/or fake reviews. These conventional approaches offer relatively accurate results for the underlying classification problem [1]. More notably, support vector machines (SVM) are frequently reported as the best classifier for a wide range of application domains. However, recent studies are demonstrating the efficacy of "ensemble learning" approaches [2], which have been shown to outperform conventional machine learning approaches such as SVM [3]. In particular, recently developed ensemble learning-based approaches such as gradient-boosting machines are demonstrating very promising results for classification.

Motivated by the recent research on ensemble learning-based approaches to classification problems [4], [5], this paper investigates and compares the performance of these ensemble-based techniques with conventional machine learning algorithms. More specifically, ensemble learning-based techniques (e.g., bagging and boosting) are integrated into conventional Support Vector Machines and extreme gradient-boosting machines and are compared with the conventional algorithms.

To accomplish this experimental study, our research team created a repository, called hereafter the "*Restaurant Dataset,*" for which a group of undergraduate students created fake reviews for three restaurants. The results of our experimental study, conducted on a repository of fake and deceptive reviews created by our research team, validate the hypothesis that the ensemble learning-based approaches outperform their conventional machine learning counterparts. The key contributions of this research are as follows:

- Introducing a new dataset on fake reviews, called the "*Restaurant Dataset*."
- Investigating the performance of ensemble learning-based approaches to the problem of identifying fake reviews.
- Reporting the results of experiments that compare the performance of ensemble learning-based approaches using different machine learning techniques for classification.

This paper is organized as follows: Section II highlights the goals of this research work and the objectives. Section III reviews the literature. A brief technical background of ensemble learning and the classifiers studied is presented in Section IV. The experimental setup of the study is discussed in Section V. The results of the experimental study are presented in Section VI. Finally, Section VII offers conclusions and sheds some light on future work.

II. Research Questions and Objectives

The objectives of this paper are two-fold: 1) introducing a newly produced dataset of fake reviews, called the "*Restaurant Dataset*," and 2) investigating the performance of ensemble learning-based methodologies in comparison with conventional machine/deep learning approaches to the fake-review detection problem. More specifically, this paper addresses the following questions:

1) Are standard forms of machine/deep learning algorithms effective in the context of fake review identification?
2) Can ensemble learning-based approaches improve the accuracy rate for this problem?

978-1-7281-7303-0/20/$31.00 ©2020 IEEE
DOI 10.1109/COMPSAC48688.2020.00-73

3) How much would the accuracy be improved through hyperparameter optimization?

III. Related work

Fuller et al. [6] developed three classification techniques – artificial neural networks, decision trees and logistic regression – along with an information fusion-based ensemble method for automated deception detection. They extracted 31 text-based deception features (cues) from the labeled statements of crime scenarios. They reported 74% accuracy with their fusion based model, 73.46% with the artificial neural network, and 71.60% with the decision tree model, respectively.

Xu et al. [7] introduced their approach on deception detection called "Unified Review Spamming Model" (URSM). The approach is a unified probabilistic graphical model for deception detection in reviews from three online services (Amazon, Yelp and TripAdvisor). By using the hierarchical Latent Dirichlet Allocation (hLDA) model [8], the model ranks the reviews, the reviewers, and the offers based on the degree of deceptivity. They compared this model with human judgments and a classifier trained using *n*-gram features. They achieved higher performance such as 78.8% accuracy for a TripAdvisor dataset compared to other methods.

Conroy et al. [9] provided a survey for deception detection in the context of implementing fake news detecting systems. They focused on two main methods: the linguistic cue approach and network analysis approach. Their study showed that the performance of linguistic approaches is topic-oriented and successful in some domains. Also, the performance of network approaches varies from average to high based on the knowledge-base which was used. They argued for an approach that combines linguistic cue and machine learning with network-based behavioral data, and explained the features for implementing such a system.

Crawford et al. [1] conducted a study on deception detection in reviews using supervised, unsupervised, and semi-supervised techniques. First, they investigated different approaches for feature extraction and divided them into "review-centric," which uses the content of the review, and "reviewer-centric," which uses review metadata related to the reviewers.

Levitan et al. [10] compared different machine learning techniques with various feature sets and proposed a hybrid deep neural network approach which uses a combination of audio and textual features (spectral, acoustic-prosodic, and lexical) for deception detection. They utilized a subset of the Columbia X-Cultural Deception (CXD) Corpus to evaluate their models and achieved F1-score of 63.90% for their deep-hybrid model and precision of 76.11% for their random forest model. In their next work [11], they examined automatic deception detection in the interview scenario. They used different sets of features: semantic features collected using LIWC, linguistic featured collected from previous study, personal traits (gender and native language) and also emotional level and detail level scores collected using Dictionary of Affect Language (DAL). Applying three different classifiers (Random Forest, Logistic Regression, and SVM), they achieved 72.74% for F1-score as the best performance by random forest.

Grondahl and Asokan [12] surveyed deception detection from textual format and provided their arguments for different questions in this subject. Based on their findings, linguistic features are not enough for textual deception detection and it is better to also consider approaches, which are based on content comparison. They also discussed adversarial "stylometry," the identification of authors based on writing style.

IV. Ensemble Learning and Classifiers

This section presents background information regarding ensemble learning and the classifiers examined in this paper.

A. Ensemble Learning

Using ensemble learning, the performance of poorly performing classifiers can be improved by creating, training, and combining the output of multiple classifiers and thus result in a more robust classification. There are three main approaches for developing an ensemble learner [13]:

- *Boosting*, often uses homogeneous-base models trained sequentially;
- *Bagging*, which often uses homogeneous-base models trained in parallel; and
- *Stacking*, which uses mostly heterogeneous-base models trained in parallel and combined using a meta-model.

By averaging (or voting) the outputs produced by the pool of classifiers, especially for small dataset, ensemble methods provide better predictions and avoid overfitting. Another reason that contributes to the better performance of ensemble learning is its ability in escaping from the local minimum. By using multiple models, the search space becomes wider and the chance for finding a better output becomes higher [14]. Ensemble learning may solve some machine learning limitations and challenges such as class imbalance, concept drift (change of features or labels over time), and curse of dimensionality.

B. Machine/Deep Learning Classifiers

We conducted our experiments using four classifiers: Decision Tree, Random Forest, Support Vector Machines (SVMs), Extreme Gradient-Boosting Trees (XGBT), and Multilayer Perceptron.

V. Experimental Setup

A. Data Collection: "Restaurant Dataset"

We developed our own fake reviews dataset (the "*Restaurant Dataset*,") in order to support research community with new dataset and thus help in addressing threats to the external validity of the experiments. The dataset consists of reviews of three restaurants, each of which has equal numbers of fake and real reviews, as well as positive and negative ones. We targeted restaurants that offered the same type of food, in order to avoid external bias. To collect these reviews, four undergraduate students wrote fake reviews (positive and negative) for three restaurants, each one paragraph in length. In addition, they collected real reviews for those three restaurants that were available from Google. To make sure that the real reviews were credible, we selected reviews from verified users. Finally, all the reviews and their labels were added to the dataset. The dataset includes 86 reviews of which 43 were fake reviews and 43 were real reviews, across all three restaurants.

B. Document Embedding for the "Restaurant Dataset"

For all classifiers in this work, we generated the features for our reviews using Doc2Vec [15]. Doc2Vec is an extension of Word2Vec. Similar to the way Word2Vec aims to represent words as vectors, Doc2Vec embeds documents as vectors [15]. In order to generate the document embedding, we preprocessed the dataset removing special characters, stop words, and punctuation marks.

C. Classification Metrics

We employed widely adopted classification metrics, such as accuracy, precision, recall, and F_1 score. An accuracy of 1.0 indicates that the predicted labels and observed labels are identical. Precision is the ratio of true positives against all predicted positive labels. Recall is the ratio of predicted true positives against all observed true positive labels. The F_1 score is the harmonic mean of precision and recall, which yields a more informative performance metric when precision and recall have dissimilar values. Although we report accuracy and F_1, we determine overall model performance using accuracy, because our balanced dataset prevents accuracy to report misleading values.

D. The Hyperparameter Optimization Algorithm

We developed Python scripts for the chosen classifiers and used the Scikit-learn library and *XGBoost* for the Extreme Gradient Boosting Trees [16]. We carried out several experiments to identify the best hyperparameters for each classifier.

Let X and y be the samples and their labels, respectively, C be a classifier with l hyperparameters to be tuned, and H be a list where the i^{th} element is either a probability distribution for the values of the i^{th} hyperparameter of C, or a list of equiprobable categorical values for the hyperparameter. A randomized search takes hyperparameter values from H for a fixed number of iterations, builds an instance of C using them, and performs the classification. In our work, we executed the randomized search 100 times, saving the model estimated in each round for future comparison. Each model was also fitted using K-fold cross-validation with $K = 10$ in each iteration. Once all iterations were completed, we selected the best model according to its accuracy, and returned both the model and its set of best hyperparameters. Our approach is defined in Algorithm 1. Algorithm 1 also returns X_{test} and y_{test}. These sets comprise unseen data and are used for the optimal classification and computing the final test errors.

The number of models estimated during the randomized search is limited to the number of iterations; however, fewer models are estimated in the event that the possible combinations of hyperparameters are fewer than the number of iterations. This procedure also prevents us from generating duplicated sets of hyperparameters.

Algorithm 1 Hyperparameters selection via randomized search

```
Input
    $X$          Samples;
    $y$          Samples labels
    $H$          Probability distributions for the $l$ hyperparameters
    $C$          Classification model
Output
    $M'$         Best model
    $H'$         Best set of hyperparameters
    $X_{test}$   Test samples
    $y_{test}$   Test samples labels

1: $X_{train}, y_{train}, X_{test}, y_{test} \leftarrow$ split_dataset$(X, y)$
2: $M = \{\}$
3: for $i \leftarrow 1$ to $100$ do
4:     $h_i \leftarrow sample(H)$
5:     $Model = Fit{-}K{-}Fold{-}CV(X_{train}, y_{train}, C, h_i)$
6:     $M \leftarrow M \cup Model$
7: end for
8: $M' = GetBestModel(M)$
9: $H' = GetParams(M')$
10: return $M', H', X_{test}, y_{test}$
```

The probability distribution over the values, or the possible categorical values, of the classifiers' hyperparameters, follows:

1) Decision Tree
   - *Max depth*: discrete uniform distribution over $[3, 5)$. This range was obtained empirically considering the size of our dataset, in order to prevent over-fitting.
   - *Splitter*: best, random.
   - *Criterion*: gini, entropy.
2) Random Forest
   - *Number of weak classifiers*: discrete uniform distribution over $[100, 1000)$.
   - *Max depth*: discrete uniform distribution over $[3, 5)$.
   - *Criterion*: gini, entropy.
3) Support Vector Machine
   - *Kernel*: RBF, linear, polynomial, sigmoid.
   - C: continuous uniform over $[0.1, 1000)$.
   - *Degree*: discrete uniform distribution over $[3, 10)$. Only considered when the kernel is polynomial.
   - *Gamma (γ)*: scale, auto. Only considered when the kernel is RBF.
4) Extreme Gradient Boosting Trees
   - *Number of stages*: discrete uniform distribution over $[100, 1000)$.
   - *Max depth*: discrete uniform distribution over $[3, 5)$.
   - *Gamma (γ)*: continuous uniform distribution over $[0, 10)$.
   - *Learning rate (α)*: continuous uniform distribution over $[0.01, 1)$.

5) Multilayer Perceptron
   - *Maximum iterations*: uniformly sampled from the set $\{600, 800, 1000, 1200\}$.
   - *Hidden layers size*: sampled from a discrete uniform distribution over $[2, 5)$. The number of units in each layer is uniformly sampled, once per layer, from $\{5, 10, 15, 20, 25, 30, 35, 40, 45, 50\}$.
   - *Activation function*: ReLU, hyperbolic tangent, logistic sigmoid, identity function.

VI. Results and discussion

Table I shows the top set of hyperparameters that result in higher accuracy during the random search for Decision Tree, Random Forest, SVM, XGBT, and MLP, respectively. We report the five best sets of hyperparameters for each classifier, except for the Random Forest, which has only three possible combinations of hyperparameters reporting a different accuracy. The accuracy and F_1 scores reported in Table I represent the mean of the estimates calculated during the 10-fold cross-validation for each model, using X_{train} and y_{train} of Algorithm 1 to fit them, so they correspond to training errors.

As Table I shows, the Decision Tree has the highest training accuracy, with 79.6%, where the "best" splitter and the Gini criterion for information gain yield the highest metrics by a margin of 7.8%. In this model, the value of maximum tree depth does not seem to make a difference in the performance.

XGBT follows in accuracy with 78.3%. The metrics for XGBT suggest smaller values for the learning rate (α), in addition to higher values of the regularization coefficient (γ), generate better accuracy and F_1 scores. Moreover, the two top results show the trade-off between maximum tree depth and the number of estimators in the ensemble, where high performance can be achieved by increasing the maximum depth with fewer numbers of estimators, or maintaining a lower depth with more estimators.

Next, SVMs report 72.6% accuracy. The RBF kernel in each set of hyperparameters suggests that the Doc2Vec representation of our dataset benefits from the non-linear mapping of the RBF kernel; this finding agrees with the high values for C, which controls the width of the soft margin in the SVM restricting samples within the margin in non-linearly separable data. γ is set to "scale" in all results, which shows that considering the variance of all our samples, in addition with the number of features to calculate γ, leads to a better accuracy. The Degree is ignored by the model, as it is used only when the kernel function is polynomial.

Random Forest reports 71.5% accuracy in training stage. The results for Random Forest show that a relatively high number of estimators achieves high accuracy independent of the maximum depth of the tree. This is consistent with the improvement in the performance of the weak classifier (decision tree) according to the number of estimators [17].

Finally, the MLP achieves 68% accuracy in training stage. As expected, the most important factor of the performance of the network is its architecture. Each value of the "Hidden layers & units" is a tuple t of positive integers, where $size(t) = \#$ of hidden layers, and the i-th element of t corresponds the number of neurons in the i-th hidden layer. The five results were generated using an MLP with 4 hidden layers, the maximum number possible in our experiments. The number of iterations and number of neurons per layer do not seem to affect the performance of the MLP, as the values for these hyperparameters range over all possible values. Regarding the activation function, ReLu and hyperbolic tangent were chosen over the rest; this agrees with the choice of the RBF kernel for the SVMs, as the ReLu and hyperbolic tangent functions allow the model to generalize in a non-linear way.

Table II shows the training and test errors for the classifiers. To report the training errors, we fit each classifier with the X_{train} and y_{train} dataset using the best set of hyperparameters presented in Table II. The test error was reported by making predictions with the classifiers using the X_{test} and y_{test} dataset. X_{train}, y_{train}, X_{test}, and y_{test} correspond to the datasets described in Algorithm 1.

We trained an ensemble of SVMs and MLPs fitted with their respective best hyperparameters. We trained both ensembles using the bagging and adaboost techniques, with the number of estimators ranging from 2 to 22 in steps of 2. Table II includes the lowest test error for the four aforementioned ensembles. For both SVM and MLP, the adaboost ensembles produce the best test error, with 0.273 for SVMs (16, 18, and 20 estimators) and 0.227 for MLPs (4 estimators).

Figures 1 and 2 show the training and testing errors for SVMs using bagging and adaboost, respectively. Figures 3 and 4 show the training and testing errors for MLPs using bagging and adaboost, respectively. The learning rates for adaboost were obtained empirically. The results of the ensemble of SVMs and MLPs suggest that the classification of Doc2Vec document embedding for the restaurant dataset benefits from adaboost over bagging. This finding might be because adaboost fits the estimators sequentially using the whole dataset, adjusting the ensemble to fit it in the best way; whereas bagging fits the estimators with different datasets that were sampled using a bootstrap from the original one [13]. This finding is consistent with the overall trend of the training error in Figure 2, where it decreases steadily as the number of estimators grows, and Figure 3, where this decreasing trend is also present, although with a few jumps. It is worth noting that the MLP adaboost ensemble fits the training data perfectly with 14 or more estimators; however, the testing error trend is less obvious for MLP ensembles. This result might be due to the limited number of samples in our dataset, as neural networks require several samples to be trained properly.

VII. Conclusion and Future Work

In this work, we present a novel dataset of fake reviews, along with the results of the binary classification using machine/deep learning techniques. Additionally, we applied ensemble learning-based approaches using Random Forest, bagging, and adaboost ensembles, with SVMs and MLPs as weak classifiers with optimized hyperparameters. Our results

TABLE I
BEST HYPERPARAMETERS TUNING (TRAINING STAGE).

Classifier	Hyperparameters				Mean	Std	Mean	Std
Decision Tree	Criterion	Max Depth	Splitter	–	Acc.	Acc.	F_1	F_1
	gini	3	best	–	**0.796**	0.097	**0.769**	0.123
	gini	4	best	–	0.772	0.115	0.751	0.137
	gini	3	random	–	0.694	0.096	0.620	0.136
	entropy	3	random	–	0.683	0.092	0.598	0.123
	entropy	4	random	–	0.657	0.095	0.576	0.148
	Hyperparameters				Mean	Std	Mean	Std
Random Forest	Criterion	Max Depth	N Estimators	–	Acc.	Acc.	F_1	F_1
	entropy	3	931	–	**0.715**	0.115	**0.673**	0.158
	gini	4	827	–	0.703	0.118	0.664	0.156
	gini	3	659	–	0.690	0.131	0.658	0.168
	Hyperparameters				Mean	Std	Mean	Std
Support Vector Machine	C	Degree	γ	Kernel	Acc.	Acc.	F_1	F_1
	21.832	6	scale	RBF	**0.726**	0.116	**0.699**	0.130
	38.581	7	scale	RBF	0.704	0.075	0.669	0.092
	33.202	5	scale	RBF	0.693	0.087	0.662	0.102
	5.404	4	scale	RBF	0.682	0.083	0.613	0.117
	4.843	9	scale	RBF	0.671	0.091	0.589	0.144
	Hyperparameters				Mean	Std	Mean	Std
Extreme Gradient Boosting Trees	γ	α	M. depth	N Estimators	Acc.	Acc.	F_1	F_1
	9.08	0.035	4	421	**0.783**	0.094	**0.726**	0.151
	8.214	0.036	3	895	0.772	0.071	0.714	0.134
	8.977	0.409	4	791	0.761	0.078	0.706	0.134
	7.144	0.231	3	544	0.750	0.082	0.697	0.140
	2.913	0.183	3	406	0.740	0.097	0.702	0.156
	Hyperparameters				Mean	Std	Mean	Std
Multilayer Perceptron	Max iterations	Hidden layers & units	Activation Function	–	Acc.	Acc.	F_1	F_1
	1000	(35, 40, 20, 5)	relu	–	**0.680**	0.100	**0.685**	0.082
	400	(5, 15, 45, 10)	tanh	–	0.671	0.120	0.682	0.107
	800	(40, 10, 20, 30)	relu	–	0.669	0.146	0.651	0.168
	600	(50, 5, 15, 45)	relu	–	0.668	0.124	0.647	0.154
	600	(10, 30, 20, 35)	tanh	–	0.667	0.147	0.678	0.135

TABLE II
TRAINING AND TEST ERRORS FOR THE CLASSIFIERS.

Classifier	**Training Error**	**Test Error**
Decision Tree	0.102	0.455
Random Forest	0.057	0.318
SVM	0.239	0.409
XGBT	0.182	0.364
MLP	0.0	0.318
Bagging Ensemble (SVM)	0.227	0.318
Adaboost Ensemble (SVM)	0.250	0.273
Bagging Ensemble (MLP)	0.034	0.318
Adaboost Ensemble (MLP)	0.068	**0.227**

show that, using document embedding from Doc2Vec and after hyperparameter optimization, stand-alone classifiers can achieve up to 68.2% accuracy in the case of MLP. Ensemble learning-based classifiers achieve up to 77.3% accuracy with the adaboost ensemble of MLPs. In every case, the ensemble of classifiers outperforms their respective base classifier, either Random Forest and XGBT with Decision Tree, or the bagging/adaboost ensembles with their respective SVMs or MLPs. Regarding the ensemble approach, adaboost seems to produce the most consistent results. As future work, we can utilize heuristics in order to find optimum values for the hyperparameters of the models. Moreover, we need to explore our method with other different datasets, and also expand the Restaurant dataset in order to make any statistical analysis based on this dataset meaningful. We also intend to perform the classification with another set of features that allows more interpretability than Doc2Vec's document embedding, and provides more linguistic insights into deceptive texts. The focus of this work was on analyzing static texts. In more challenging settings, when there are interactions between two parties (e.g., chatting and lie detection), more sophisticated analysis would be needed to take into account the time dimension (i.e., a time series problem) such as Long Short-Term Memory [18], [19] and clustering using deep learning [20], reinforcement learning [21], and evidence theory [22].

ACKNOWLEDGMENT

Thanks to Pritish Ayer, Sagar Lamichhane, Omer Qureshi, and Pranaya Sharma for contributions to the Restaurant dataset creation. This research work is supported by National Science Foundation under Grant No: 1723765.

REFERENCES

[1] M. Crawford, T. M. Khoshgoftaar, J. D. Prusa, A. N. Richter, and H. Al Najada, "Survey of review spam detection using machine learning techniques," *Journal of Big Data*, vol. 2, no. 1, 2015.

[2] T. Dietterich, "Ensemble methods in machine learning," in *1st International Workshop on Multiple Classifier Systems*, 2000, pp. 1–15.

[3] A. Statnikov, L. Wang, and C. Aliferis, "A comprehensive comparison of random forests and support vector machines for microarray-based cancer classification," *BMC Bioinformatics*, vol. 9, no. 319, 2008.

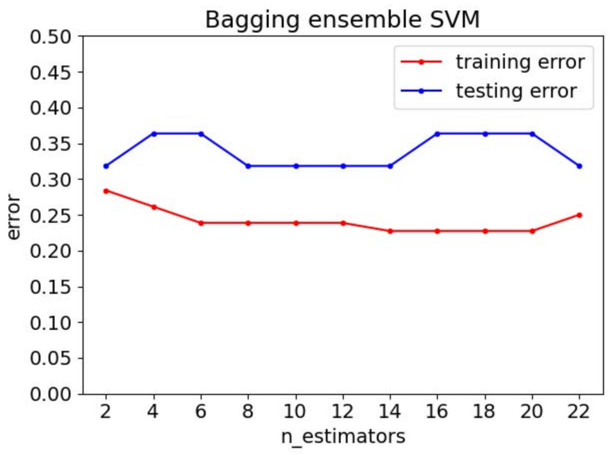

Fig. 1. Training and test error for different number of estimators in the bagging ensemble for SVMs.

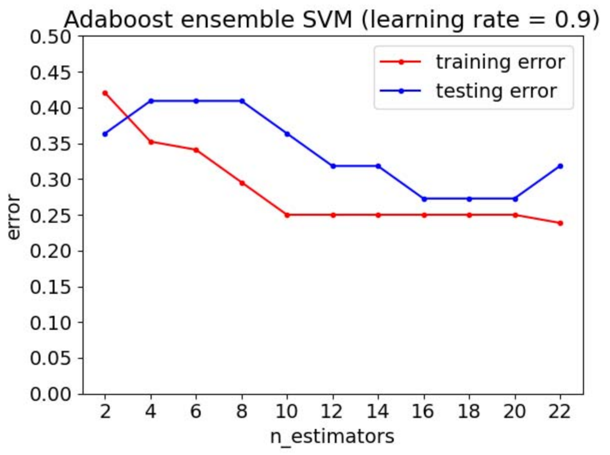

Fig. 2. Training and test error for different number of estimators in the adaboost ensemble for SVMs.

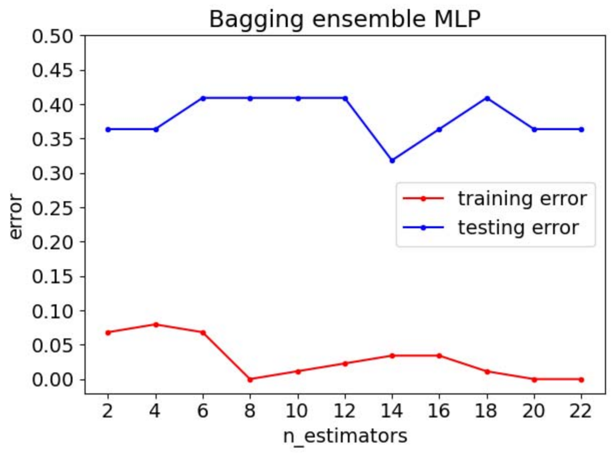

Fig. 3. Training and test error for different number of estimators in the bagging ensemble for MLPs.

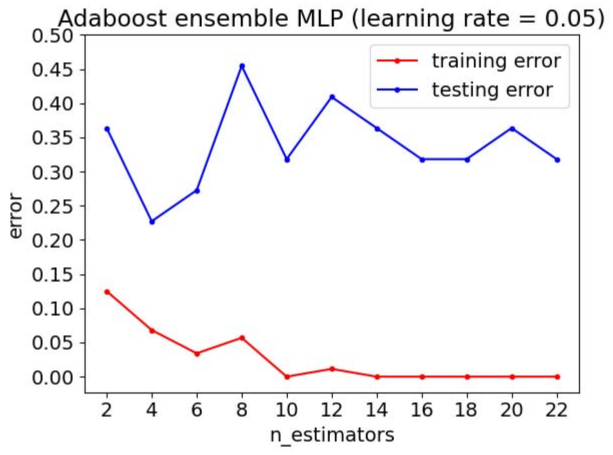

Fig. 4. Training and test error for different number of estimators in the adaboost ensemble for MLPs.

[4] Y. Pang, X. Xue, and A. S. Namin, "Early identification of vulnerable software components via ensemble learning," in *IEEE ICMLA*, 2016.

[5] F. Abri, S. Siami-Namini, M. A. Khanghah, F. M. Soltani, and A. S. Namin, "Can machine/deep learning classifiers detect zero-day malware with high accuracy?" in *IEEE Big Data*, 2019, pp. 3252–3259.

[6] C. Fuller, D. Biros, and D. Delen, "An investigation of data and text mining methods for real world deception detection," *Expert Syst. Appl.*, vol. 38, pp. 8392–8398, 07 2011.

[7] Y. Xu, B. Shi, W. Tian, and W. Lam, "A unified model for unsupervised opinion spamming detection incorporating text generality," in *24th International Conference on Artificial Intelligence*, 2015, pp. 725–731.

[8] D. M. Blei, T. L. Griffiths, and M. I. Jordan, "The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies," *J. ACM*, vol. 57, no. 2, pp. 7:1–7:30, 2010.

[9] N. Conroy, V. Rubin, and Y. Chen, "Automatic deception detection: Methods for finding fake news," in *Proceedings of the Association for Information Science and Technology*, 2016.

[10] G. Mendels, S. I. Levitan, K.-Z. Lee, and J. Hirschberg, "Hybrid acoustic-lexical deep learning approach for deception detection," in *Proc. Interspeech*, 2017, pp. 1472–1476.

[11] S. I. Levitan, A. Maredia, and J. Hirschberg, "Linguistic cues to deception and perceived deception in interview dialogues," in *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, 2018.

[12] T. Grondahl and N. Asokan, "Text analysis in adversarial settings: Does deception leave a stylistic trace?" *ACM Comput. Surv.*, vol. 52, no. 3, pp. 45:1–45:36, Jun. 2019.

[13] Z.-H. Zhou, "Ensemble learning," in *Encyclopedia of Biometrics*, 2009.

[14] O. Sagi and L. Rokach, "Ensemble learning: A survey," *Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, 2018.

[15] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *ICML*, 2014, pp. 1188–1196.

[16] T. Chen, T. He, M. Benesty, V. Khotilovich, and Y. Tang, "Xgboost: extreme gradient boosting," *R package version 0.4-2*, pp. 1–4, 2015.

[17] L. Breiman, "Random Forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[18] S. Siami-Namini, N. Tavakoli, and A. S. Namin, "A comparison of ARIMA and LSTM in forecasting time series," in *IEEE ICMLA*, 2018.

[19] ——, "The performance of LSTM and BiLSTM in forecasting time series," in *IEEE Big Data*, 2019, pp. 3285–3292.

[20] N. Tavakoli, S. Siami-Namini, M. A. Khanghah, F. M. Soltani, and A. S. Namin, "An autoencoder-based deep learning approach for clustering time series data," *Springer Nature (SN) Applied Sciences*, 2020.

[21] M. Chatterjee and A. S. Namin, "Detecting phishing websites through deep reinforcement learning," in *IEEE COMPSAC*, 2019, pp. 227–232.

[22] ——, "Detecting web spams using evidence theory," in *COMPSAC*, 2018, pp. 695–700.