On Comparative Classification of Relevant Covid-19 Tweets

Gokhan Bakal
Computer Engineering Department
Abdullah Gul University
Kayseri, Turkey
Email: gokhan.bakal@agu.edu.tr

Orhan Abar
Computer Engineering Department
Osmaniye Korkut Ata University
Osmaniye, Turkey
Email: orhanabar@osmaniye.edu.tr

2021 6th International Conference on Computer Science and Engineering (UBMK) | 978-1-6654-2908-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/UBMK52708.2021.9558945

***Abstract*—Due to the impressive information dissemination power of social networks such as Twitter, people tend to check social networks and Web pages more than other traditional news sources, including newspapers, TV news programs, or radio channels. In that sense, the information carried by the content of the shared social media posts becomes much more considerable. However, most of the posts are commonly either irrelevant or inaccurate. Besides, the more critical case than the correctness of the information is the diffusion speed on Twitter through the reply or retweet actions. These activities make the initial situation even more complicated than itself due to the unregulated nature of the social networks and the lack of an immediate verification mechanism for the correctness of the posts. When we consider the current Covid-19 pandemic period (causing the coronavirus disease), one of the most utilized information resources is Twitter except the official health administration institutions. Thereupon, examining the correctness of the information related to the Covid-19 pandemic by computational techniques (e.g., Data Mining, Machine Learning, and Deep Learning) has been gaining popularity and remains a substantial task. Hence, we mainly focused on analyzing the correctness of the posts related to the current pandemic shared on the Twitter platform. Therefore, the overall goal of this work is to classify the relevant tweets using linear and non-linear machine learning models. We achieved the best F1 performance score (99%) with the neural network model using the unigram features & threshold value of 50 among all model configurations.**

***Index Terms*—natural language processing, tweet classification, text mining, machine learning**

I. Introduction

People usually want to spread their thoughts on a topic as much as possible due to their nature. To parallel this fact, having technological advancements in almost every part of our lives also transformed our opinion-sharing habits. Social media applications have emerged as principle sharing platforms in the last two decades. From this aspect, Twitter is one of the leading social media platforms used by people across the world. The essential feature that makes Twitter the mainstream opinion sharing platform compared to others is the retweeting (reshare/repost) mechanism. This retweeting feature allows people to disseminate tweets they like by just clicking a single retweet button. Therefore, the Twitter platform has become a critical data resource to understand/analyze the emergent thoughts of society on a particular topic of interest.

Specifically, during extraordinary periods, such as natural disasters, global disease outbreaks, etc., the size of daily produced data dramatically increased. However, this outcome causes another difficulty, disinformation, which is the diffusion action of intentionally false information to confuse people on a global scale. Hence, we employed various linear and non-linear models utilizing extracted conventional textual features (n-grams) to classify relevant tweet examples. According to the results we got, we achieved the best performance measures with the configuration having the lowest threshold value of t ($t \in \{50, 100, 250, 500, 1000, 2500\}$), which controls the size of sub-feature space determined by the occurring frequency counts of the feature elements in the dataset.

II. Background & Related Works

Social media applications became gigantic data sources for specific research efforts across multiple disciplines since they have consecutively emerged and been accepted by society fast for more than two decades [1]–[5].

One of the widely studied topics on Twitter data is sentiment analysis which evolved into a more specific emotion analysis task [6], [7]. For this purpose, Agarwal et al. [2] introduced fundamental sentiment analysis techniques, including rule-based, machine learning, and hybrid algorithms. The central motivation of sentiment analysis is to measure/understand what people think about a particular subject. Another popular research field on Twitter data is to process health-related tweets. Sutton et al. [8] examined the structural details of lung cancer related content posted by individuals and organizations to Twitter. Similarly, Bakal and Kavuluru [9] demonstrated the diffusion patterns of the Twitter posts collected by thematic keywords for the medical domain. As a distinct study field, people also worked on shared posts to predict political views during the election periods. Specifically, Wang et al. [10] built a model that functions real-time on opinion analysis for the 2012 U.S. Presidential Election. Beyond, the most critical research effort on Twitter is to identify relevant or genuine tweets among the irrelevant/fake ones due to the diffusion speed of Twitter posts. Bovet et al. [11] illustrated how fake news on Twitter disseminate during the 2016 U.S. presidential election using graph-based algorithms. In addition, Meda et al. [12] extracted 13 informative features to discover

978-1-6654-2908-5/21/$31.00 ©2021 IEEE

spamming accounts on Twitter. Then, the authors employed traditional machine learning algorithms, such as Random Forest, Decision Tree, AdaBoost, and LogitBoost classifiers to discover the spammers. Since we experiment with an extraordinary period all over the world due to the coronavirus epidemic, we intended to analyze a Covid-19 related dataset, that is generated by collecting tweets containing specific keywords. In this work, we focused to build classification models **to separate true/relevant tweets from misleading tweets** posted during a particular period.

III. Dataset Curation & Stats

A typical tweet post is a digital representation that consists of a maximum of 280 characters long textual and possibly some non-textual objects, including user mentions, hashtags, URLs, images, videos as well as emoticons. Although non-textual objects indicate a specific meaning, we executed the computational analysis only on the textual parts of the tweet. We admit that there are multiple publicly available Twitter datasets regarding the Covid-19 pandemic period. Nevertheless, we prefer to curate our dataset since we filter the tweet stream by a broader set of predefined keywords as {*2019-nCoV, nCoV19, corona, covid, covid19, covid_19, covid-19, covid 19, virus, coronavirus, corona_virus, corona-virus, SARS-COV-2, covid-19 virus, coronanews, Covid-19 live updates, epidemic, vaccine, pandemic, face mask*}. Another point is that *Twitter streaming API* captures a broad range of tweet posts. For instance, we are able to collect the tweets if the URL in the content links to a webpage, the title of which has any of the keywords. Here, the practical advantage of this ability is that we will also have contextually relevant tweets even if they do not have the keywords.

In this work, we build our main objective in this specific data collection aspect to obtain more realistic and strong negative tweet examples. In total, we gathered 6,022,326 tweets (including individual retweets) satisfying our keywords during the period between July 10, 2020 and July 20, 2020. One primary problem of tweet analysis studies for tweet data is the reliability of the content due to the lack of instant verification. The other issue is the high diffusion speed of the posts due to the follower-followee mechanism. Yet another problem is whether the captured content is relevant to the research topic. To tackle these problems, we filtered our dataset by particular rules. First, we obtained only the tweets that comprise at least one URL entity. Second, we subsequently kept only the tweets that have at least a 100 retweet count. This filtration step is necessary since we aim to aggregate more informational tweets to some extend. Eventually, we collected 71,517 tweets that include at least one of the predefined keywords and a URL as our positive/relevant data examples. Plus, we also have a distinct set of tweets where each one has a URL entity but not a keyword in the textual part. By this knowledge, we accumulated 30,510 tweets containing at least one URL as the negative/irrelevant examples. We deliberately created this negative dataset because we made our models enforced to predict the correct labels for the hard-to-classify test set created by the collection rules we applied.

A. Data Cleaning

As we stated in Section III, each tweet has a maximum of 280 characters long textual data along with non-textual data items, such as images, videos, emoticons, emojis, etc. We investigate the textual part only to avoid potential inconsistencies and noises. To illustrate that better, consider the case in which we have two tweet examples conveying the same meaning by the textual data. However, despite having similar textual knowledge, these tweet examples may comprise an opposite type of emoticons causing semantic contradiction. Hence, we applied a set of data cleaning steps as below.

- We first remove all emoji items an d emoticons appearing in the tweet statuses,
- Before applying additional filtering operations, we utilized a publicly available library "*pyspellchecker*" [13] to convert misspelled words if appear any,
- We also filtered URLs, user mentions, and retweet indicators (as a prefix "RT") which are basically non-descriptive and noise prune entities in our models,
- Then, we take the punctuation symbols and stop words off,
- At the final stage, to extract the most descriptive features over the filtered tweets, we exploit a fast and powerful Natural Language Processing (NLP) tool, *SpaCy* [14], which is open-source and publicly available.

IV. Methods

In this section, we start with the introduction of how we extracted the features to use them in the models and the model configurations in detail.

A. Textual Feature Extractions

In many textual data studies, the most informative features are utilizing n-grams features extracted from the preprocessed/cleaned text documents. Besides, the n-gram features are extracted n consecutive words/tokens (where $n > 0$, typically between 1 and 5) over the documents [15]. To generate the n-gram feature vectors, we use the frequency counts of the n-grams in the documents/tweets. In usual document analysis, the overpopulated feature space is pruned by a predefined threshold value. This case is because we can both enhance the model performance by reducing the less descriptive feature elements and have a more efficient experimental setup. To better understand the pruning effect over the feature space, we used a set of threshold values {2500, 1000, 500, 250, 100, 50} when we build the models.

B. Machine Learning Models

The overall motivation of this study is to classify the Covid-19 related tweets into relevant and irrelevant subsets. For the classification task, one effective way is to utilize supervised learning approaches. The purpose of supervised learning is to learn from pre-labeled training examples and predict the

correct labels for unseen test instances. Here, our concentration is binary classification, where the aim is to build machine-learned models, which can have higher generalization ability gained from the training dataset, to assign a label, $y \in \{0, 1\}$, to each example in test dataset.

Each example is represented by a *n* dimensional feature vector **x =** $(x_1, .., x_n)$, where each feature x_i indicates a distinct piece of predictive information. To create this feature representation, instances in the dataset are forced to be transformed from the raw input space (e.g. raw textual documents/tweets) into the *n* dimensional feature space.

To perform the classification task with the aim of comparative analysis, we exploited different ML algorithms, including Logistic Regression (LR), XGBoost, and fully connected feed-forward neural network (FFNN) algorithms. For each experimental setup, we have five n-gram configurations (*uni-gram, bi-gram, tri-gram, four-gram, and five-gram*) as well as six threshold configurations we expressed in the previous section. Consequently, we have 30 different essential model setups. Plus, we randomly shuffled the curated dataset ten times to create ten distinct datasets to disclose the generalization power for the models and to derive average performance scores. So, we split each shuffled dataset into three subsets; 70% as the training set, 20% as the test set, and the remaining 10% of the dataset as the validation set. As a high-level graphical representation of this study is illustrated in Figure 1. We briefly mentioned each ML algorithm in the following consecutive parts.

1) Logistic Regression: LR is a widely used supervised machine learning algorithm [16]. It has the logistic/sigmoid function that transforms/fits the input space values to the interval of [0, 1] that also helps us to handle the probabilistic evaluations. Even though it is mostly known for binary classification problems [17], it can be used for multi-label classification problems with the one-vs-rest fashion, too. In our study, we used the LR algorithm available through the Python scikit-learn [18] for binary classification. To identify the best-performing model, we used the validation dataset along with a grid search-based approach over the LR parameters, such as penalty-regularization (penalty $\in$ {"l1, "l2"}) type, regularization strength ($c \in 0.01, 0.1, 1, 10$), and solver algorithm (solver $\in \{$"$liblinear$", "$saga$", "$lbfgs$"$\}$) in the model.

2) XGBoost: XGBoost is an ML algorithm using a gradient boosting infrastructure. It is technically a decision-tree-based approach. Nowadays, neural network-based models are powerful options compared to the traditional ML models [19], in particular on unstructured high-volume text data. However, the XGBoost algorithm has been recently gaining great attention owing to considerable classification performance, efficiency, and providing more interpretable outcomes compared to neural network models. We used publicly available XGBoost python package [20] in the experiments. Similarly, we subjected the validation dataset to the XGBoost model to select optimum parameters, including booster (booster $\in \{$"$gblinear$", "$gbtree$", "$dart$"$\}$), eta (also known as learning rate $\in \{0.001, 0.01, 0.1\}$), maximum depth (max_depth $\in \{6, 8, 10\}$), and objective (objective $\in \{$"$squarederror$", "$hinge$"$\}$) in the experiments.

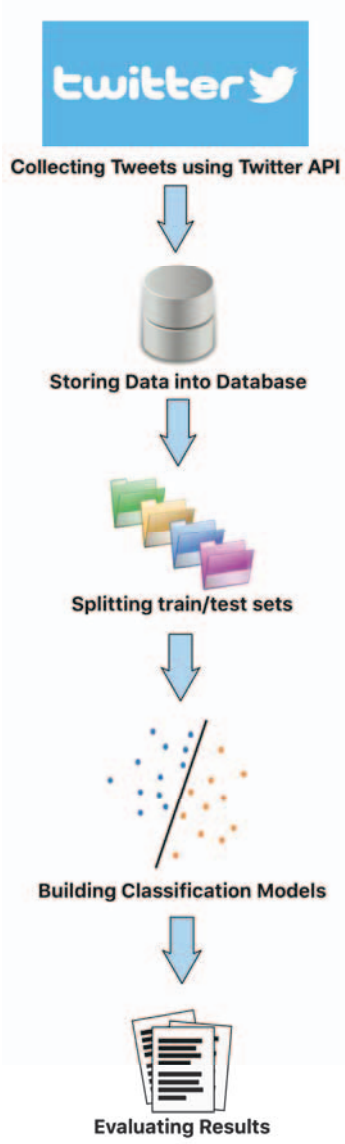

Fig. 1. Overall methodology diagram

3) Fead-Forward Neural Network: Neural Network (NN) algorithms are known as the methods which imitate human neural systems to process data inputs as training. So, a typical neural network is built with inter-connected items called neurons (artificial). Neural networks can have single or multiple layers of artificial neurons. For example, Multi-Layer Perceptron (MLP) is a typical feed-forward neural network algorithm and constructed with a minimum of three layers of nodes: *an input layer*, *a hidden layer*, and *an output layer*. Each layer has several processing units, and each unit is fully connected with weighted connections to the ones in the following layer.

$$\mathcal{L} = -(y \log(p) + (1 - y) \log(1 - p)) \quad (1)$$

TABLE I
CONSOLIDATED PERFORMANCE SCORES FOR ALL MODEL CONFIGURATIONS

		Logistic Regression			**XGBoost**			**Feed-Forward NN**		
	Threshold	***Precision***	***Recall***	***F1-score***	***Precision***	***Recall***	***F1-score***	***Precision***	***Recall***	***F1-score***
Unigram	2500	0.9914	0.9948	0.9931	0.9958	0.9952	0.9955	0.9969	0.9976	0.9972
	1000	0.9973	0.9983	0.9978	0.9985	0.9992	0.9988	0.9992	0.9981	0.9987
	500	0.9990	0.9994	0.9992	0.9985	0.9992	0.9988	0.9993	0.9985	0.9989
	250	0.9991	0.9995	0.9993	0.9988	0.9993	0.9991	0.9994	0.9986	0.9990
	100	0.9990	0.9994	0.9992	0.9988	0.9994	0.9991	0.9993	0.9987	0.9990
	50	0.9993	0.9996	0.9995	0.9993	0.9996	**0.9995**	**0.9994**	0.9987	**0.9991**
Bigram	2500	0.8383	0.5030	0.6287	0.8240	0.5037	0.6253	0.7023	0.9998	0.8250
	1000	0.7485	0.5410	0.6280	0.7702	0.5391	0.6342	0.7205	0.9889	0.8336
	500	0.7756	0.6052	0.6799	0.7763	0.6027	0.6786	0.7512	0.9822	0.8513
	250	0.7988	0.8542	0.8256	0.8304	0.6961	0.7573	0.9588	0.7717	0.8551
	100	0.8816	0.9292	0.9047	0.8818	0.9302	0.9053	0.9903	0.8769	0.9301
	50	0.9281	0.9612	0.9443	0.9274	0.9612	0.9440	0.9933	0.9274	**0.9593**
Trigram	2500	0.3504	0.5	0.4121	0.3504	0.5	0.4121	0.7009	1.0	0.8242
	1000	0.3504	0.5	0.4121	0.3504	0.5	0.4121	0.7009	1.0	0.8242
	500	0.8572	0.5315	0.6562	0.8572	0.5315	0.6562	0.7144	0.9999	0.8334
	250	0.8573	0.5527	0.6721	0.8587	0.5527	0.6726	0.7237	0.9997	0.8396
	100	0.8724	0.6241	0.7277	0.8726	0.6240	0.7276	0.7559	0.9983	0.8603
	50	0.8851	0.6832	0.7712	0.8848	0.6830	0.7710	0.7858	0.9974	**0.8790**
Fourgram	2500	0.3504	0.5	0.4121	0.3504	0.5	0.4121	0.7009	1.0	0.8242
	1000	0.3504	0.5	0.4121	0.3504	0.5	0.4121	0.7009	1.0	0.8242
	500	0.8572	0.5315	0.6562	0.8572	0.5315	0.6562	0.7144	0.9999	0.8334
	250	0.8589	0.5489	0.6696	0.8593	0.5486	0.6697	0.7213	0.9999	0.8381
	100	0.8697	0.5922	0.7046	0.8697	0.5922	0.7046	0.7408	0.9998	0.8510
	50	0.8813	0.6398	0.7414	0.8813	0.6399	0.7414	0.7637	0.9998	**0.8659**
Fivegram	2500	0.3504	0.5	0.4121	0.3504	0.5	0.4121	0.7009	1.0	0.8242
	1000	0.3504	0.5	0.4121	0.3504	0.5	0.4121	0.7009	1.0	0.8242
	500	0.3504	0.5	0.4121	0.3504	0.5	0.4121	0.7009	1.0	0.8242
	250	0.8448	0.5102	0.6361	0.8448	0.5102	0.6361	0.7047	0.9999	0.8268
	100	0.8550	0.5329	0.6566	0.8550	0.5329	0.6566	0.7148	0.9999	0.8337
	50	0.8658	0.5771	0.6925	0.8658	0.5771	0.6925	0.7345	0.9999	**0.8469**

As we did the same with the LR and XGBoost models, we utilized the validation set to achieve the best performance results by fine-tuning the models. Considering the hyperparameter tuning job, we used the *RandomSearch* function available in the Keras platform. For this purpose, we tried to optimize the number of units in the hidden layer (nodes $\in \{16, 32, 48, 64\}$), activation functions ("$relu$", "$tanh$", "$sigmoid$"), and learning rate (learning_rate $\in \{0.001, 0.01, 0.1\}$) in the NN models. In the training phase, we employed *binary cross-entropy* loss function as shown in Equation 1 with *Adam* optimizer and a batch size of 32 with 20 epochs.

V. RESULTS & DISCUSSION

Consequently, we targeted to classify Covid-19 tweets into relevant and irrelevant as a binary classification task. To that end, we built distinct ML models using both traditional and NN-based algorithms. As we mentioned in Section IV-B, we conducted experiments with ten randomly shuffled datasets to derive the average performance metrics† for each model. In Table I, we show the performance scores for the models with multiple n-grams and threshold configurations. When we look at the result table, the first noticeable outcome is that we gained an improvement trend for the F1 measure when we decrease the threshold value of all distinct configurations. This result is not surprising because we make the models have more discriminative textual features. Besides, using a high threshold value makes the feature space more sparse; so, the models can not learn the dataset properly to distinguish the test instances. Another critical point for the models is that employing higher n-grams leads to having worse performance. For example, we achieved the best performance scores with unigram models regardless of the algorithms used. This outcome is unquestionable because the tweet content is not long enough to contain higher n-grams (indicating bigram, trigram, four-gram, and five-gram) features. Additionally, we realized that Logistic Regression and XGBoost models yield quite a similar performance for each experiment although both of them are completely different techniques. Unlike Logistic Regression and XGBoost models, we obtained a solid achievement with the Feed-Forward NN model when we engage longer n-grams even though there is a decreasing trend for the F1 measure. Considering the differences between the best performing bigram and trigram models, there is around a 28% drop in recall measure for both Logistic Regression and XGBoost models, while the Feed-Forward NN model instead improved the recall measure by nearly 7%. Yet another differentiation between the models is that the linear models barely outperform the Feed-Forward NN model in the unigram model with the feature space threshold of 50, unlike the rest of the model configurations. One potential explanation of this particular case is that linear models have better classification performances compared to the neural network models when

†We computed the final average F1-score by using average precision and average recall scores.

the feature space is notable huge. As a final remarkable point, we noticed that the optimum n-gram type apart from the unigram type is bigram. This circumstance is because, beyond the bigram model, performance metrics are getting dramatically decreased. Another logical reason is that the cleaned tweet examples can not trigger corresponding higher n-gram features due to the lower word counts.

VI. Conclusion

In this work, we intended to build an automation system that can identify the relevant tweets on the Covid-19 pandemic. To that end, we generated a tweet collection infrastructure to generate our dataset. Using the collected data examples, we built Logistic Regression, XGBoost, and Feed-Forward NN models by multiple n-gram features (where $n \in \{1, 2, 3, 4, 5\}$) along with six feature space limitation thresholds. Comparatively, we demonstrated n-gram textual features are powerful descriptors for the classification task. The results also proved that Feed-Forward NN models outperformed traditional ML models for the majority of the distinct model configurations.

References

[1] F. Benevenuto, G. Magno, T. Rodrigues, and V. Almeida, "Detecting spammers on twitter," in *Collaboration, electronic messaging, anti-abuse and spam conference (CEAS)*, vol. 6, no. 2010, 2010, pp. 75–83.

[2] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, and R. J. Passonneau, "Sentiment analysis of twitter data," in *Proceedings of the workshop on language in social media (LSM 2011)*, 2011, pp. 30–38.

[3] J. Parker, Y. Wei, A. Yates, O. Frieder, and N. Goharian, "A framework for detecting public health trends with twitter," in *Proceedings of the 2013 IEEE/ACM international conference on advances in social networks analysis and mining*, 2013, pp. 556–563.

[4] D. Scanfeld, V. Scanfeld, and E. L. Larson, "Dissemination of health information through social networks: Twitter and antibiotics," *American journal of infection control*, vol. 38, no. 3, pp. 182–188, 2010.

[5] G. Bakal, H. Abar, I. Ozturk, and O. Abar, *Text Mining Applications Using Real-World Data in Python*. Ankara: Nobel Press, February 2021.

[6] A. Kumar and T. M. Sebastian, "Sentiment analysis on twitter," *International Journal of Computer Science Issues (IJCSI)*, vol. 9, no. 4, pp. 372–378, 2012.

[7] X. Wang, F. Wei, X. Liu, M. Zhou, and M. Zhang, "Topic sentiment analysis in twitter: a graph-based hashtag sentiment classification approach," in *Proceedings of the 20th ACM international conference on Information and knowledge management*, 2011, pp. 1031–1040.

[8] J. Sutton, S. C. Vos, M. K. Olson, C. Woods, E. Cohen, C. B. Gibson, N. E. Phillips, J. L. Studts, J. M. Eberth, and C. T. Butts, "Lung cancer messages on twitter: content analysis and evaluation," *Journal of the American College of Radiology*, vol. 15, no. 1, pp. 210–217, 2018.

[9] G. Bakal and R. Kavuluru, "On quantifying diffusion of health information on twitter," in *2017 IEEE EMBS International conference on biomedical & health informatics (BHI)*. IEEE, 2017, pp. 485–488.

[10] H. Wang, D. Can, A. Kazemzadeh, F. Bar, and S. Narayanan, "A system for real-time twitter sentiment analysis of 2012 us presidential election cycle," in *Proceedings of the ACL 2012 system demonstrations*, 2012, pp. 115–120.

[11] A. Bovet and H. A. Makse, "Influence of fake news in twitter during the 2016 us presidential election," *Nature communications*, vol. 10, no. 1, pp. 1–14, 2019.

[12] C. Meda, F. Bisio, P. Gastaldo, and R. Zunino, "A machine learning approach for twitter spammers detection," in *2014 international carnahan conference on security technology (iccst)*. IEEE, 2014, pp. 1–6.

[13] P. Lison and J. Tiedemann, "Extracting large parallel corpora from movie and tv subtitles," in *Proceedings of the Language Resources and Evaluation Conference (LREC)*, 2016, pp. 923–929.

[14] M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd, "spaCy: Industrial-strength Natural Language Processing in Python," 2020. [Online]. Available: https://doi.org/10.5281/zenodo.1212303

[15] H. Ahmed, I. Traore, and S. Saad, "Detection of online fake news using n-gram analysis and machine learning techniques," in *International conference on intelligent, secure, and dependable systems in distributed and cloud environments*. Springer, 2017, pp. 127–138.

[16] D. G. Kleinbaum and M. Klein, *Logistic Regression: A Self-Learning Text*, ser. Statistics for Biology and Health. Springer New York, 2010.

[17] G. Bakal, P. Talari, E. V. Kakani, and R. Kavuluru, "Exploiting semantic patterns over biomedical knowledge graphs for predicting treatment and causative relations," *Journal of biomedical informatics*, vol. 82, pp. 189–199, 2018.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[19] S. Jain, A. K. Jain, and S. P. Singh, "Building a machine learning model for unstructured text classification: Towards hybrid approach," in *Rising Threats in Expert Applications and Solutions*. Singapore: Springer, 2021, pp. 447–454.

[20] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: ACM, 2016, pp. 785–794.