Hyperparameter Tuning of Semi-Supervised Learning for Indonesian Text Annotation

Siti Khomsah[1], Nur Heri Cahyana[2], Agus Sasmito Aribowo[3]
Department of Data Science, Institut Teknologi Telkom Purwokerto, Indonesia[1]
Department of Informatics, Universitas Pembangunan Nasional Veteran Yogyakarta, Indonesia[2, 3]

***Abstract*—A crucial issue in sentiment analysis primarily relies on the annotation task involving data labeling. This critical step is typically performed by linguists, as the nuanced meaning of text significantly influences its contextual interpretation. If there is a large volume of data, annotation is time-consuming and financially burdensome. Addressing these challenges, a semi-supervised learning annotation (SSL) that integrates human annotator and artificial intelligence algorithms emerges as a potent solution. Building accurate SSL needs to explore the best architecture, including a combination of machine learning and mechanism. This research aims to construct semi-supervised model annotation text by tuning the parameter of the machine learning algorithm to gain the most accurate model. This study employed a Support Vector Machine and a Random Forest algorithm to build semi-supervised annotation. Grid-Search and Random-Search were employed to tune the Random Forest and Support Vector Machine parameters. The semi-supervised annotation model was applied to annotate Indonesian texts. The outcomes signify that hyperparameter-tuning enhances SSL performance, surpassing the performance achieved using default parameters. The experiment also shows that the SSL annotation using a Support Vector Machine tuned by Grid Search and Random Search is more robust than the Random Forest algorithm. Hyperparameter tuning is also robust to training data that contains many manual labeling errors by experts.**

Keywords—Text annotation; semi-supervised; parameter-tuning; grid search; random search

I. Introduction

The challenge in text classification-based machine learning is data labeling. Each textual instance has meaning upon the context and grammatical nuance specific to each language. Thus, human intervention is vital for data labeling, as humans are adept at assessing the contextual relevance of text. Human annotator requires expertise in understanding the language context. However, labeling numerous volumes of data requires quite an amount of time and causes tiredness for the annotator - consequently, the objectivity loss of the labeling process. Hence, automation annotation is highly needed before the dataset feeds into a machine learning classifier.

There has been a new development of an annotator-based machine built by learning knowledge of the language experts or humans. SSL uses a small sample of data annotated by language experts and then uses the sample to build a training model. Then, the training model is used for the annotation of unlabeled data. The SSL model draws expert insight from a small sample to build robust annotator unlabeled data. Here, the SSL challenge is to produce a reliable and precise model. In previous related research, a semi-supervised learning (SSL) model was developed to classify text [1-4]. The performance of semi-supervised text annotation still needs to improve. Al-Laith et al. used LSTM and FastText for annotating Arabic text with only three classes, resulting in an accuracy of 69.4% on the SemEval 2017 dataset. In contrast, the best system achieved a 63.38% F1 score, while on the ASTD dataset, performance improved from 64.10% to 68.1% [4]. Aydln and Güngör [5] combined semi-supervised and unsupervised methods and implemented the J-48 tree, Support Vector Machine, and Naive Bayes algorithms. As a result, the accuracy is more than 90%. However, this model has not proven its performance for multi-class datasets and other language datasets. Alahmary and Al-Dossari [2] applied Naive Bayes as a semi-supervised learning classifier with high accuracy (83%). However, researchers have yet to test this model on other datasets, and we do not know its performance when utilized for SSL in other languages. So, developing a semi-supervised text annotator for Indonesian text that utilizes multiple machine learning models and various vectorizers is expected to yield improved performance. Previous related Research SSL in Indonesian by NurHeri Cahyana et al. annotates hate speech [3]. The maximum accuracy in those research employing KNN and TF-IDF is only 59.68%. SSL model in [3] has a weakness: the model has not been applied to other Indonesian text datasets and has not tried other model combinations.

Many ways are done to gain a high-performance model, such as identifying data samples with hesitant labels and removing training data with unreliable labels. Those can improve the quality of the training data and improve classifier performance [6]. Another approach to improve model performance is to overcome by trying various amounts of training data proportions when building the annotation model, then applying the ensemble method, using several machine learning to classify the same data consensus on classification decisions using confidence values [7]. Performance machine learning depends on parameter setting [8]. Some ways to leverage the performance of machine learning are tuning the parameters [9—11] and applying optimization [12-14]. The goal of tuning these parameters is to find the best combination parameter [15]. Tuning parameters can significantly impact the model performance and finding the right combination involving experimentation and iterative adjustments [16]. A tuning parameter is a parameter that is not learned directly from the data during the training process of a machine-learning algorithm. Instead, it is a value set before training to control the algorithm behavior. Unlike the default individual parameters of a model, which are learned from the training data, tuning parameters are set by the user before training. Fine-tuning parameters can have a substantial impact on the model performance [17].

Besides tuning individual algorithm parameters, hyperparameter techniques such as Grid Search and Random Search can help find the best parameter combination. Several researchers in text labeling or other domains employ Grid Search [9], [18], [19] and Random Search [10], [11], [20], [21] to enhance the accuracy and performance of machine learning algorithm. Not all parameter tuning techniques are in tune with machine learning algorithms and the data they handle. In previous research, several algorithms that can generally be tuned include Support Vector Machine (SVM), Random Forest (RF), Logistic Regression, Naive Bayes, Neural Networks, and Gradian Boosting. It is necessary to research the best hyperparameters for each algorithm. This research aims to leverage the performance of semi-supervised text annotation through tuning parameters Support Vector Machine and Random Forest using Grid Search and Random Search. Different language corpus needs an appropriate architecture model for automated text labeling. This research uses the Indonesian dataset for the research material. Thus, SSL architecture for the Indonesian dataset becomes the focus.

Our research proposed SSL with the SVM and Random Forest as classifier. Random Search and Grid Search as hyperparameter tuning to obtain the most optimal model. The research problem boundary employs hyperparameter tuning to gain high-performing SSL models for Indonesian text annotation—the performance metric using F1-Score and accuracy. This research is limited to utilizing two machine learning algorithms, Random Forests and Support Vector Machine, and two hyperparameters technic Grid Search and Random Search. We divide the article into four sections: introduction, methods, results and discussion, and conclusions.

II. Methods

The following section describes the research steps, including data collection, data cleaning, feature extraction, building classifier model annotation, and evaluation.

A. Data Collection

Due to several reasons, this research has used various datasets to test the Semi-Supervised Text Annotation Model. The primary rationale behind this approach is to enhance the model's ability to generalize and effectively accommodate variations within the data. By subjecting the model to evaluation using diverse datasets, the model can objectively assess its performance across different domains. Also, this approach can examine the model's efficacy in managing linguistic heterogeneity. Additionally, testing on various datasets has comprehensively evaluated the model performance. Thus, this research endeavor involves the utilization of three publicly available datasets, as described in Table I, for the explicit purpose of rigorous testing and analysis.

TABLE I. DATASETS USED FOR MODEL ASSESSMENT

No	Dataset	Instance	Source
1.	Hate Speech	13168	https://github.com/okkyibrohim/ id-multi-label-hate-speech-and-abusive-language-detection/blob/master/re_dataset.csv
2	Sentiment Ridife	10805	https://github.com/ridife/dataset-idsa/blob/master/ Indonesian%20Sentiment%20Twitter% 20Dataset%20Labeled.csv
3	IndoNLU Sentiment	12759	https://github.com/IndoNLP/IndoNLU/tree/master/dataset/ smsa_doc-sentiment-prosa

Before their utilization in the testing phase, the four datasets were processed by the same steps, including data cleaning, word embedding, and feature extraction. The subsequent section elaborates on the empirical outcomes of analyzing these four distinct datasets.

B. Data Cleaning and Preprocessing

The datasets used in this experiment comprised comments in the Indonesian language. The cleaning purpose is to clear up the dataset from noise such as punctuation, numerical character, and stop words. The clean data was transformed into a vector through several stages, including tokenization (unigram, bigram, and trigram), stemming, and finally, turning the stem into the vector using TF-IDF. Tokenizing onto unigram is breaking down a piece of a sentence into individual units. Bigram is a pair of sequence words within a sentence, while trigram refers to three sequence words within a sentence. Unigram, bigram, and trigram are N-gram types in which *N* is any number. N-Gram with *N* is two or more, often used to capture more contextual information than a single word.

C. Word Embedding and Feature Extraction

The Bag of Words (BoW), often mentioned as Term-Frequency (TF), constitutes an algorithm employed to determine the weight of individual words within a document. The weight of Term-Frequency is computed by quantifying the occurrence of the term *t* in document *D* and dividing it by the total count of words present in document *D*. The underlying objective is to identify unique words that can serve as an essential document feature. The large documents generate a big matrix. Given a number feature of *N* and a sum document of *D*, the feature matrix has dimensions of $N \ x \ D$. TF is the occurrence of a word *t* in document *D*, computed as in Eq. (1).

$$TF_{t,d} = \frac{n_{t,d}}{Totalnumber\ of\ terms\ in\ document} \qquad (1)$$

The $TF_{t,d}$ is the frequency of term *t* in document *d*, where *t* is a term (word within a sentence), *f* is the number of occurrences of term *t* in document *d*, and $n_{t,d}$ is the number of terms *t* in document *d*.

TF-IDF (Term Frequency-Inverse Document Frequency) is the weight computed by multiplication between TF and IDF. IDF (Inverse Document Frequency) is a value that indicates how important a word is in the entire document in the dataset. A word with a lower TF-IDF value is considered less important, and vice versa. Words that appear frequently throughout the document are considered as less important words. The weight IDF of a document calculated as in Eq. (2)

$$IDF_d = log(\frac{Number\ of\ Document}{Number\ of\ document\ with\ term\ t}) \quad (2)$$

To compute the TF-IDF (Term Frequency-Inverse Document Frequency), the TF value is multiplied by the IDF value according to the following formula, as in Eq. (3)

$$TFIDF_{t,d} = tf_{t,d}\ x\ idf_d \quad (3)$$

D. Model of Semi-Supervised Text Annotation

Random Forest is a robust ensemble learning algorithm widely used in machine learning for classification. Random Forest defines the target class by combining multiple decision tree outputs to yield a single outcome. As suggested by its name, a "forest" comprises numerous trees generated through bagging or bootstrap aggregating. Each tree in the Random Forest produces class predictions, with the majority class prediction as the candidate prediction model. Increasing the number of trees leads to enhanced accuracy and mitigates overfitting concerns. Random Forest is known for its robustness and resistance to overfitting. Tuning its parameters can improve performance on specific datasets.

SVM is a robust machine learning algorithm widely used for classification tasks. It is particularly effective for tasks involving complex data distribution and when clear dividing lines are needed to differentiate between classes. SVM aims to find the optimal hyperplane that best separates different classes of data points in a high-dimensional space. The basic idea of SVM is to find a hyperplane that maximizes the margin between classes of data points. The margin is defined as the distance between the hyperplane and the nearest data points from each class. The idea is to choose the hyperplane with the largest margin, which is expected to classify well to new data with no class yet. However, SVM's effectiveness depends on properly tuning parameters to make the data separable.

E. Proposed Semi-Supervised Learning Architecture

This research proposes a novel architecture for the semi-supervised learning (SSL) model with tuning parameters, depicted in Fig. 1. The SSL workflow initiates by utilizing an annotated dataset encompassing training, testing, and unlabeled data, as in Fig. 1. Training and testing data are manually labeled by an Indonesian language expert.

The process begins with word embedding, transforming textual data from the training set into vectors using the TF-IDF technique. This word embedding procedure generates three distinctive vectors: unigram, bigram, and trigram. These three vector representations subsequently serve as inputs for building three separate models employing Random Forest and Support Vector Machine (SVM). The algorithm Random Forest and SVM parameters were tuned to gain the best classification model.

Following the stacking principle, these three distinct models operate independently. Each model participates in the annotation of the unlabeled data. Then, the unlabeled data is annotated by each model, resulting in pseudo-labels of three sets of datasets, each classified according to one of the three models. A pseudo-label has high confidence if the cumulative weight assigned to it divided by the cumulative weight of all models is high. This confidence value is compared to a predefined threshold. This threshold serves as a criterion to identify annotated data (with pseudo-labels) with confidence values deserving inclusion as part of the training data. Then, documents with high-ranking confidence and existing training data are mixed. Through this innovative approach, the SSL process harnesses the strengths of multiple models to improve predictions on unlabeled data iteratively. Considering confidence values and applying the threshold optimizes integrating pseudo-labeled data into the training set, ultimately enhancing the model's performance.

Fig. 2 describes the main algorithm of SSL. The process started by reading the labeled training dataset (DT), testing dataset (DTest), and unlabeled dataset (UN). All three datasets are transformed into the vector with feature unigrams, bigrams, and trigrams. Then, the model-building process is shown in lines 9-11, the hyperparameter process is in lines 12-14, the annotation process with the best parameter model is in lines 15-17, and the line 18-21 is the voting process.

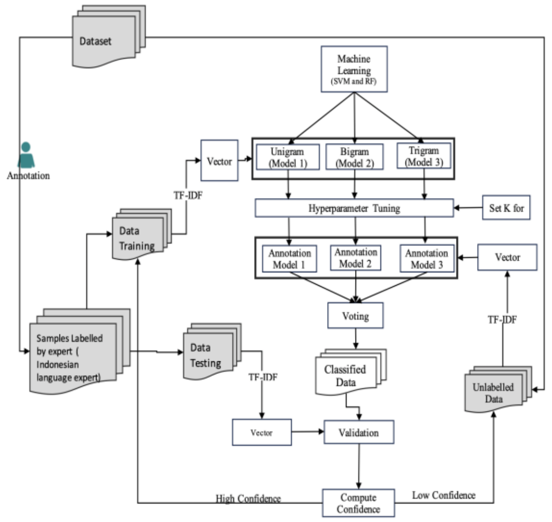

Fig. 1. SSL text annotation using parameter tuning.

```
1   DEF SSL(X,ML,HyP):
2   READ DT      // Data Training(X,y)
3   READ DTest   // Data Testing(X,y)
4   READ UN      // Unlabeled Data(X)
5   VTestUni, VTestBi, VTestTri =TFIDF (DTest, ngram=1,2,3)
6   Loop three times or until no unlabelled dataset :
7       VTrainUni, VTrainBi, VTrainTri = TFIDF(DT, ngram=1,2,3)
8       VUnalbUni, VUnlblBi, VUnlblTri =TFIDF(UN, ngram=1,2,3)
9       Mod1 = ML.Train(VTrainUni)
10      Mod2 = ML.Train(VTrainBi)
11      Mod3 = ML.Train(VTrainTri)
12      Accuracy1,bestparam1=Hyperparam(Mod1,HyP)
13      Accuracy2,bestparam2=Hyperparam(Mod2,HyP)
14      Accuracy3,bestparam3=Hyperparam(Mod3,HyP)
15      Label[1]=Mod1.PredictProba(VUnlblUni, bestparam1)
16      Label[2]=Mod2.PredictProba(VUnlblBi, bestparam2)
17      Label[3]=Mod3.PredictProba(VUnlblTri, bestparam3)
18      For J = 1 to LEN(UN):
19          Newlabel[J], WeightLabel[J]=Voting(Label[1,2,3].RecNo[J]
20          Move(UN[J],Newlabel[J]) to DT
21  Output(DT)
22  Validate(DT) with Accuracy, F1Score
23  END
24
25  BEGIN
26  ML=['RF', 'SVM']
27  Hyperparameter=['GridSearch','RandomSearch'])
28  For X in ML:
29      DO SSL(X, ML,Hyperparameter)
28  END
```

Fig. 2. Algorithm of proposed SSL.

F. Tuning Techniques

Grid Search and Random Search were applied for tuning both SVM and RF. *Grid Search* is a hyperparameter tuning technique that systematically searches for a machine learning optimal combination model of hyperparameter values. It involves defining a grid of possible hyperparameter values and then evaluating the model performance using each combination of these values through cross-validation. The best parameter combination selected is the optimal set of hyperparameters [18].

Random Search does not explore all combinations like Grid Search. Instead, it tracks the combinations that provide the best performance to date. When more combinations are evaluated, the combination becomes the new best combination if the performance of a combination is better than the previous one [22]. This randomness can be more efficient in exploring the hyperparameter space, primarily when enormous search space exists.

G. Evaluation Model

To evaluate the classification performance of the SSL model, we employ a confusion matrix, as shown in Table II. The confusion matrix will compare the predicted results with the actual class using the rules in Table II. This study uses two parameters for model validation, namely accuracy and F1-score.

TABLE II. CONFUSION MATRIX

		Actual	
		Positive	Negative
Predicted	Positive	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative(TN)

Accuracy is the ratio of the correctly predicted dataset to all datasets in the experiment. Accuracy, as in Eq. (4), is a good measurement, but it is only on symmetric datasets, i.e., when the number of false positives and false negatives is almost the same or balanced.

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \quad (4)$$

Precision measures the proportion of genuinely positive instances among all instances predicted as positive by the model. In other words, it quantifies how well the model avoids false positives. High precision indicates that when the model predicts a positive class, it is more likely to be correct. Precision is crucial where the instances of false positives are high or when we want to ensure that the positive predictions made by the model are accurate. However, it is worth noting that precision does not consider instances that were predicted as negative, which could be actual positive cases that were missed (false negatives). Eq. (5) is the precision formula.

$$Precision = \frac{TP}{TP+FP} \quad (5)$$

Recall (sensitivity) is the opposite of precision, as in Eq. (6).

$$Recall = \frac{TP}{TP+FN} \quad (6)$$

Eq. (7) is the average weight of precision and recall. F1 Score is more beneficial than accuracy, especially if the results have an unequal class distribution.

$$F1\,Score = \frac{2(Recall * Precision)}{Recall + Precision} \quad (7)$$

III. RESULT AND DISCUSSION

A. Data Distribution

The following Table III shows the distribution of class data for each dataset. Ridife corpus has class positive (24%), neutral (49.1%), and negative (26.9%). IndoNLU Sentiment has class positive (57.7%), neutral (10.7%), and negative (31.6%). Hate Speech only has two classes: positive (42.2%) and negative (57.8%). There are all three datasets in imbalanced class data.

TABLE III. DISTRIBUTION LABEL CLASS

Datasets	Label Class						
	Positive	Proportion	Neutral	Proportion	Negative	Proportion	Total
Ridife	2574	24.0%	5271	49.1%	2882	26.9%	10727
IndoNLU Sentiment	7359	57.7%	1367	10.7%	4034	31.6%	12760
Hate Speech	5561	42.2%	-	-	7606	57.8%	13167

The proposed SSL model employed a tuning parameter. Our experiment uses two techniques, namely Random-Search and Grid-Search. Random Forest and SVM parameters were tuned to gain the best performance of the SSL model. Before applying the SSL model, all datasets corpus are dispart into training, testing, and unlabeled data. Training and testing data are each 10% of the dataset—human labels 10% of training and testing data. The remaining 90% is unlabeled data.

B. Performance SSL Model

Data testing was employed to assess the performance of the SSL model under both baseline and final conditions. Baseline conditions involved the SSL model being constructed solely using labeled training data. In contrast, the final condition entailed forming the SSL model by fusing labeled training data and Pseudo-Labels generated from unlabeled training data. The experiment was done in two distinct stages. The initial stage entailed evaluating the SSL model using hyperparameters Grid Search, while the subsequent stage involved testing the SSL model performance with hyperparameters using Random Search.

1) Performance SSL using SVM: The performance of SSL without hyperparameter is shown in Table IV, while after tuning is shown in Tables V and VI.

TABLE IV. PERFORMANCE SSL SVM

Datasets	Without Hyperparameter			
	Accuracy (%)	Precision (%)	Recall (%)	F1-Score(%)
Ridife	26.9	100	26.9	42.3
IndoNLU Sentiment	57.6	100	57.6	73.1
Hate speech	57.8	100	57.8	73.2

TABLE V. PERFORMANCE SSL SVM TUNED BY GRID SEARCH

Datasets	Grid Search			
	Accuracy (%)	Precision (%)	Recall (%)	F1-Score(%)
Ridife	50.0	97.8	50.0	65.2
IndoNLU Sentiment	78.4	85.5	78.4	81.4
Hate speech	73.7	82.6	73.7	75.6

TABLE VI. PERFORMANCE SSL SVM TUNED BY RANDOM SEARCH

Datasets	Random Search			
	Accuracy (%)	Precision (%)	Recall (%)	F1-Score(%)
Ridife	50.0	98.0	50.0	65.3
IndoNLU Sentiment	78.8	85.2	78.8	81.3
Hate Speech	73.0	83.5	73.0	75.4

Table IV shows the result of the SSL model using SVM with standard parameters having low performance. Accuracy towards all three datasets is under 60%, even though the F1-Score is high. The F1 score shows that the SSL SVM model without parameter tuning is quite good, although only for some data (IndoNLU sentiment and Hate Speech). The Ridife dataset does not reach 50% accuracy, requiring further investigation of the condition of the data, especially the validity of the sample annotation results by Indonesian language experts. With perfect precision values, it is surprising that the recall values are so much lower. With these results, an analysis can be drawn that the standard parameters used in SVM cannot optimize the performance of the SSL model.

Meanwhile, Table V shows that SVM tuning using Grid Search can increase accuracy, indicated by increased accuracy and F1 scores. Because of the differences in class distribution, the performance observations emphasize the F1 score. By comparing the performance of the models without tuning and with tuning, Grid Search increases the F1 Score in the three datasets by 22.9% in the Ridife dataset, 8.3% in the IndoNLU sentiment dataset, and 2.4% in the Hate Speech dataset. Random Search increased the F1 Score by 23.0%, 8.2%, and 2.2%, respectively, on Ridife, IndoNLU sentiment, and Hate speech.

The effect of the parameter tuning done for SVM can be seen in the F1 score performance, as shown in Fig. 3. The F1 score is a valid evaluation to represent the unequal class distribution. Fig. 3 shows that both tuning techniques (Grid and Random Search) can improve the SSL model performance.

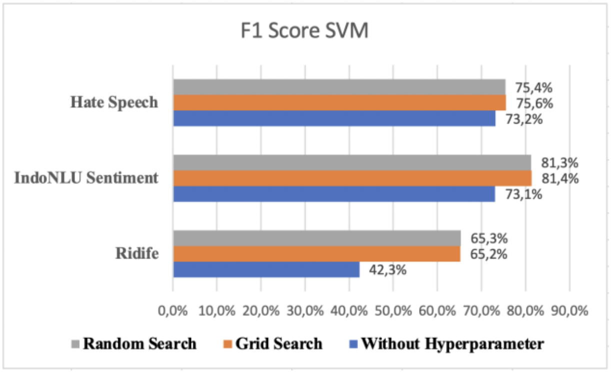

Fig. 3. Performance F1 Score SVM.

2) *Performance SSL using random forest (RF)*: The performance of SSL RF without a hyperparameter is shown in Table VII; SSL tuned by Grid is in Table VIII, and tuned by Random Search is in Table IX. In contrast, tuning on RF does not improve the SSL model's performance like tuning on SVM.

TABLE VII. PERFORMANCE SSL RF

Datasets	Without Hyperparameter			
	Accuracy (%)	Precision (%)	Recall (%)	F1-Score(%)
Ridife	31.5	75.2	31.5	37.1
IndoNLU Sentiment	75.0	81.0	75.0	76.4
Hate Speech	72.8	85.0	72.8	75.5

TABLE VIII. PERFORMANCE SSL RF TUNED BY GRID SEARCH

Datasets	Grid Search			
	Accuracy (%)	Precision (%)	Recall (%)	F1-Score(%)
Ridife	46.0	60.0	46.0	51.0
IndoNLU Sentiment	75.4	80.9	75.4	76.7
Hate Speech	72.6	86.2	72.6	75.6

TABLE IX. PERFORMANCE SSL RF TUNED BY RANDOM SEARCH

Datasets	Random Search			
	Accuracy (%)	Precision (%)	Recall (%)	F1-Score(%)
Ridife	36.5	71.2	36.5	40.7
IndoNLU Sentiment	75.6	81.1	75.6	76.9
Hate Speech	72.0	85.7	72.0	75.0

Table VII shows that the SSL model using RF with standard parameters performs poorly for the Ridife dataset, which only reached 40.7%. However, the IndoNLU Sentiment and Hate Speech datasets are pretty good, above 75%. Meanwhile, Table VII shows that tuning parameter RF using Grid Search was unsuccessful enough to increase model performance. It can be seen that F1 Score before and after tuning in the IndoNLU Sentiment and Hate Speech datasets. By comparing the performance of the RF model without tuning and tuning with Grid Search, the improving performance in all datasets is 13.9% for the Ridife dataset: 0.3% for the IndoNLU Sentiment dataset, and 0.1% for the Hate Speech dataset. While tuning with Random Search could not improve RF performance significantly. The effect of Random Search tuning on RF is only shown by the Ridife dataset. Random Forest is a tree-based algorithm that uses an ensemble tree to increase performance. Therefore, setting up RF with Random Search did not work significantly. The graphical visualization in Fig. 4 supports this. Suppose the conditions of the initial data are observed in more detail. In that case, the Ridife dataset tends to have a lot of noise, slang words, and inaccurate labeling by experts, which may be good for examining the effect of tuning. Meanwhile, the other two datasets (IndoNLU Sentiment and Hate Speech) are relatively cleaner. Considering the condition of the dataset, tuning is suitable for models built from a lot of noise-training data.

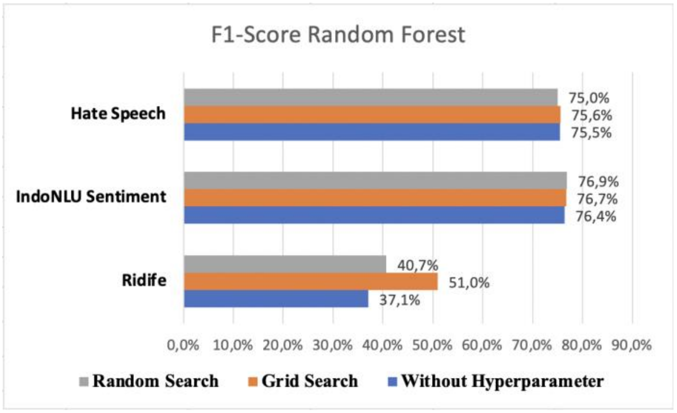

Fig. 4. Performance F1 score RF.

We find that our SSL model for the hate speech dataset obtained higher accuracy (more than 75%) than the SSL proposed by Nur Heri Cahyana et al. [6], which has only reached an accuracy of 59.68% using KNN. Our proposed method proved that SVM and Random Forest perform better for hate speech datasets. In initial data conditions, the Ridife dataset contains more noise, slang words, and inaccurate labels from experts. In contrast, the IndoNLU Sentiment and Hate speech datasets have less noise, and expert labeling is precise. This research found that the proposed SSL using hyperparameter tuning is more suitable for noisy datasets. Hyperparameter tuning is also robust to training data that contains many manual labeling errors by experts.

IV. Conclusion

Annotation or data labeling in sentiment analysis is a substantial stage in the case of numerous large datasets. Annotation is time-consuming if humans do it. Thus, building model annotation using a computer is needed, but the accuracy model is notable. This research uses a semi-supervised model for annotating sentiment using a Support Vector Machine (SVM) and a Random Forest (RF) algorithm. SVM and RF were respectively tested as classifiers. To gain the most accurate model, RF and SVM were tuned using Random-Search and Grid-Search, respectively. The experiment used three Indonesian corpora as a dataset (Ridife, IndoNLU Sentiment, and Hate speech). Overall, Grid-Search and Random Search leverage performance only in the Ridife dataset. The result shows that tuning works significantly on SVM, but on RF, it does not work on all datasets. This research found that models with hyperparameter tuning are robust to training data containing a lot of noise and incorrect human labeling. For further experiments, employing many variation datasets and paying attention to imbalanced and noise conditions in the data are suggested.

Acknowledgment

The authors are grateful to The Ministry of Education, Culture, Research and Technology, Indonesia, for funding this research through the Fundamental Research Grant 2023, which has led to the publication of this paper.

References

[1] V. L. S. Lee, K. H. Gan, T. P. Tan, and R. Abdullah, “Semi-Supervised Learning for Sentiment Classification using Small Number of Labeled Data,” in The Fifth Information Systems International Conference, Surabaya: Elsevier B.V., 2019, pp. 577–584. doi: 10.1016/j.procs.2019.11.159.

[2] R. Alahmary and H. Al-Dossari, “A semiautomatic annotation approach for sentiment analysis,” *J Inf Sci*, 2021, doi: 10.1177/01655515211006594.

[3] N. H. Cahyana, S. Saifullah, Y. Fauziah, A. S. Aribowo, and R. Drezewski, “Semi-Supervised Text Annotation for Hate Speech Detection using K-Nearest Neighbors and Term Frequency-Inverse Document Frequency,” *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 10, pp. 147–151, 2022.

[4] A. Al-Laith, M. Shahbaz, H. F. Alaskar, and A. Rehmat, “Arasencorpus: A semi-supervised approach for sentiment annotation of a large arabic text corpus,” *Applied Sciences (Switzerland)*, vol. 11, no. 5, Mar. 2021, doi: 10.3390/app11052434.

[5] C. R. Aydln and T. Güngör, “Sentiment Analysis in Turkish: Supervised, Semi-Supervised, and Unsupervised Techniques,” 2021. doi: 10.1017/S1351324920000200.

[6] K. Miok, G. Pirs, and M. Robnik-Sikonja, “Bayesian Methods for Semi-supervised Text Annotation,” 2020. [Online]. Available: http://arxiv.org/abs/2010.14872

[7] N. H. Cahyana, S. Saifullah, Y. Fauziah, A. S. Aribowo, and R. Drezewski, “Semi-Supervised Text Annotation for Hate Speech Detection using K-Nearest Neighbors and Term Frequency-Inverse Document Frequency,” *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 10, pp. 147–151, 2022.

[8] H. J. P. Weerts, A. C. Mueller, and J. Vanschoren, “Importance of Tuning Hyperparameters of Machine Learning Algorithms,” Jul. 2020, [Online]. Available: http://arxiv.org/abs/2007.07588

[9] R. Ghawi and J. Pfeffer, “Efficient Hyperparameter Tuning with Grid Search for Text Categorization using kNN Approach with BM25 Similarity,” *Open Computer Science*, vol. 9, no. 1, pp. 160–180, 2019, doi: 10.1515/comp-2019-0011.

[10] L. Villalobos-Arias, C. Quesada-López, J. Guevara-Coto, A. Martínez, and M. Jenkins, “Evaluating hyper-parameter tuning using random search in support vector machines for software effort estimation,” in *PROMISE 2020 - Proceedings of the 16th ACM International Conference on Predictive Models and Data Analytics in Software Engineering, Co-located with ESEC/FSE 2020*, Association for Computing Machinery, Inc, Nov. 2020, pp. 31–40. doi: 10.1145/3416508.3417121.

[11] R. Turner *et al.*, “Bayesian Optimization is Superior to Random Search for Machine Learning Hyperparameter Tuning: Analysis of the Black-Box Optimization Challenge 2020,” in *Proceedings of Machine Learning Research*, 2021, pp. 3–26.

[12] A. Nugroho and H. Suhartanto, “Hyper-Parameter Tuning based on Random Search for DenseNet Optimization,” in *7th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*, IEEE Xplore, 2020, pp. 96–99. doi: 10.1109/ICITACEE50144.2020.9239164.

[13] L. Yang and A. Shami, “On hyperparameter optimization of machine learning algorithms: Theory and practice,” *Neurocomputing*, vol. 415, pp. 295–316, 2020, doi: 10.1016/j.neucom.2020.07.061.

[14] E. S. Tellez, D. Moctezuma, S. Miranda-Jiménez, and M. Graff, “An automated text categorization framework based on hyperparameter optimization,” *Knowl Based Syst*, vol. 149, pp. 110–123, 2018, doi: 10.1016/j.knosys.2018.03.003.

[15] Y. Xie, C. Zhu, W. Zhou, Z. Li, X. Liu, and M. Tu, “Evaluation of machine learning methods for formation lithology identification: A comparison of tuning processes and model performances,” *J Pet Sci Eng*, vol. 160, pp. 182–193, Jan. 2018, doi: 10.1016/j.petrol.2017.10.028.

[16] E. Elgeldawi, A. Sayed, A. R. Galal, and A. M. Zaki, “Hyperparameter Tuning for Machine Learning Algorithms Used for Arabic Sentiment Analysis,” *Informatics*, vol. 8, no. 79, pp. 1–21, 2021.

[17] Md Riyad Hossain and Douglas Timmer, "Machine learning model optimization with hyper-parameter tuning approach," in *International Conference on Advanced Engineering, Technology and Applications (ICAETA)*, 2021. [Online]. Available: https://www.researchgate.net/publication/354495368

[18] S. George and B. Sumathi, "Grid Search Tuning of Hyperparameters in Random Forest Classifier for Customer Feedback Sentiment Prediction," 2020. [Online]. Available: www.ijacsa.thesai.org

[19] B. H. Shekar and G. Dagnew, "Grid search-based hyperparameter tuning and classification of microarray cancer data," in *2019 2nd International Conference on Advanced Computational an Communication Paradigms (CACCP)*, IEEE, 2019, pp. 1–8. doi: 10.1109/ICACCP.2019.8882943.

[20] R. Turner *et al.*, "Bayesian Optimization is Superior to Random Search for Machine Learning Hyperparameter Tuning: Analysis of the Black-Box Optimization Challenge 2020," in *Proceedings of Machine Learning Research*, 2020, pp. 3–26.

[21] R. G. Mantovani, A. L. D. Rossi, J. Vanschoren, B. Bischl, and A. C. P. L. F. De Carvalho, "Effectiveness of Random Search in SVM hyper-parameter tuning," *Proceedings of the International Joint Conference on Neural Networks*, vol. 2015-Septe, 2015, doi: 10.1109/IJCNN.2015.7280664.

[22] S. Andradóttir, "A Review of Random Search Methods," in *Handbook of Simulation Optimization*, M. C. Fu, Ed., New York, NY: Springer New York, 2015, pp. 277–292. doi: 10.1007/978-1-4939-1384-8_10.