ISSN: 2834-4928

Research Article

Journal of Electrical Electronics Engineering

Tackling Sexism in Social Media: Multilingual AI Solutions

Ron Keinan*

Department of Computer Science, Jerusalem College of Technology, Lev Academic Center, Israel

***Corresponding Author**
Ron Keinan, Department of Computer Science, Jerusalem College of Technology, Lev Academic Center, Israel.

Submitted: 2024, Sep 05; **Accepted**: 2024, Oct 01; **Published**: 2024, Oct 30

Citation: Keinan, R. (2024). Tackling Sexism in Social Media: Multilingual AI Solutions. *J Electrical Electron Eng, 3*(5), 01-07.

Abstract
In this paper, we describe our submission to the EXIST-2024 contest. We participated in Task 1: "Sexism Identification in Tweets" in both English and Spanish. To classify the tweets for sexist content, we developed various models by altering the machine learning classifier, feature type (word/character n-grams), feature quantity, and text preprocessing steps. We then vectorized the text using the TF-IDF embedding technique. After training these configurations on the training dataset, we selected the best models based on accuracy and F1-score on the development set and used them to predict the test labels. Our top-performing model achieved an F1 score of 72.23, securing 39th place out of 70 participants.

Keywords: Sexism Identification, Machine Learning, TF-IDF, Feature Selection, Char Based n-Grams

1. Introduction
The identification of sexism in social networks has become a critical challenge in the field of Natural Language Processing (NLP). Detecting and classifying sexist content in social media posts is crucial for promoting respectful and inclusive online environments. This task holds significance not only for platforms managing content but also for addressing broader societal concerns, such as curbing the spread of harmful stereotypes and promoting gender equality [1].

Social networks have become central platforms for activism and global movements, including MeToo, 8M, and Time'sUp. These movements have empowered women worldwide to share their experiences of abuse, discrimination, and sexism. While social media amplifies these voices, it also serves as a breeding ground for sexism and other forms of disrespectful behavior [2]. As such, developing automated tools for detecting sexism in social networks is crucial. These tools can assist in identifying and flagging sexist content in real-time, contributing to content moderation and enabling an understanding of how sexism manifests in online discourse.

In this context, we describe our participation in the EXIST-2024 contest where we focused on Task 1: "Sexism Identification in Tweets" in English and Spanish [3,4]. Our approach involved constructing multiple models by varying several components: the machine learning classifier, the type of features (word or character n-grams), the quantity of features, and the preprocessing applied to the text data. We vectorized the text using the Term Frequency-Inverse Document Frequency (TF-IDF) embedding technique.

The relevance of this task is heightened by the increasing volume of user-generated content on social media platforms, where rapid identification and intervention in cases of sexist content can greatly influence the safety and user experience. By combining preprocessing, feature extraction, and various machine learning techniques, our approach adds to ongoing efforts in creating robust systems for detecting sexist language.

1.1 Background
1.1.1 Feature Extraction
Feature selection plays a crucial role in text classification tasks, as it helps improve model performance by identifying the most informative elements from the text. In our approach to sexism identification, we focused on selecting features based on two primary types: word n-grams and character n-grams.

1.1.2 Word and Character N-grams
Word n-grams refer to sequences of words that appear together in the text, capturing syntactic structures and contextual relationships. By using sequences of words, n-grams enable models to better grasp the semantic meaning carried by word combinations. For instance, in a bigram model, pairs of consecutive words are analyzed, while a trigram model examines sequences of three words. Using the sentence "The quick brown fox jumps over the lazy dog," bigrams would include "The quick," "quick brown," "brown fox," and so

on, while trigrams would cover "The quick brown," "quick brown fox," and so forth.

However, word n-grams have limitations, particularly when dealing with sparse data and out-of vocabulary words, which are common in social media text. To address these challenges, we applied TF-IDF weighting to emphasize rare but informative n-grams and downplay more common but less useful ones.

Character n-grams, particularly those with word boundaries (char-wb), divide text into sequences of characters while preserving word boundaries. This technique excels in capturing morphological patterns and managing text variations such as typos, slang, and informal language that are common in social media. For example, character n-grams of length six from the word "identification" might include "identi," "dentif," and "entifi." The inclusion of word boundaries allows the model to maintain word integrity while learning from character-level patterns.

Our experiments showed that character n-grams of medium length (around six characters) consistently outperformed word n-grams. This suggests that character n-grams are better suited for capturing the nuanced morphological features and informal expressions typical of sexist language. Their ability to handle different linguistic forms and idiomatic expressions was particularly beneficial for our dataset, which included a diverse range of colloquial sexist remarks.

1.1.3 Comparative Analysis
Through extensive experimentation, we found that models using character n-grams with word boundaries achieved higher accuracy and F1 scores than those using word n-grams alone. This demonstrates the effectiveness of character n-grams in capturing subtle and context-dependent expressions of sexism that may not be detected by word-level features.

1.1.4 TF-IDF Embeddings
To optimize feature selection, we used the Term Frequency-Inverse Document Frequency (TF-IDF) technique. TF-IDF helps to quantify the significance of each n-gram by balancing its occurrence within a document against its frequency across the dataset, emphasizing informative features that contribute to classification.

1.1.5 Text Embeddings
Text embeddings are a method for representing text in a continuous vector space, which allows algorithms to process and analyze text data effectively. These embeddings capture both semantic and syntactic similarities between words or documents, aiding a variety of NLP tasks, such as sentiment analysis, document classification, and information retrieval.

1.1.6 Types of Text Embeddings
There are different types of text embeddings, each suited to specific tasks:

- Word Embeddings Word embeddings, like Word2Vec and GloVe, map individual words to high dimensional vectors based on their contextual use. Words with similar meanings, such as "king" and "queen," are positioned close together in the vector space. Word embeddings are highly effective for tasks that require understanding word semantics.
- Contextualized Word Embeddings Contextualized word embeddings, generated by models like ELMo, BERT, and GPT, provide representations that adapt to a word's context in a sentence. Unlike static embeddings, these models can capture the different meanings of polysemous words, such as "bank" in the phrases "bank of a river" and "banking institution." This context sensitivity makes them especially useful for tasks like named entity recognition and machine translation.
- Document Embeddings Document embeddings extend the concept of word embeddings to longer text units, such as sentences or paragraphs. Techniques like Doc2Vec and Universal Sentence Encoder provide fixed-length vectors that represent the entire text, useful for document classification and clustering tasks.

1.1.7 Significance in NLP
Text embeddings represent a significant advancement in NLP by providing dense and continuous text representations, which traditional bag-of-words models lack. Embeddings handle large vocabularies efficiently and capture intricate relationships between words, greatly improving the performance of various NLP tasks.

1.1.8 TF-IDF Embeddings
In this study, we applied the TF-IDF (Term Frequency-Inverse Document Frequency) embedding technique to convert text into numerical form for model training [5].

TF-IDF is a widely used method that calculates the importance of words by combining their frequency within a document with their rarity across the entire dataset. Words with high TF-IDF scores are considered more informative for the classification task.

The TF-IDF (Term Frequency-Inverse Document Frequency) score is calculated as follows:

$$\text{TF-IDF}(t,d,D) = \text{TF}(t,d) \times \text{IDF}(t,D) \qquad (1)$$

Where:

$$\text{TF}(t,d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d}$$

$$\text{IDF}(t,D) = \log\left(\frac{\text{Total number of documents in the corpus } |D|}{\text{Number of documents containing term } t}\right)$$

By employing these diverse embedding techniques, we aimed to capture the rich semantic and syntactic features of the text, enhancing the performance of our models in identifying and classifying sexist content in social media posts.

1.2 Machine Learning Classifiers
In our approach, we experimented with several machine learning classifiers to identify the most effective model for sexism

identification. Each classifier brings specific strengths, and the following are the key classifiers we employed:

- Random Forest Classifier (RandomForestClassifier): An ensemble learning method that constructs multiple decision trees and outputs the majority vote, known for high accuracy and resistance to overfitting [6,7].
- Extra Trees Classifier (ExtraTreesClassifier): An ensemble model that builds unpruned decision trees from random subsets of the data, enhancing model robustness [8].
- LightGBM Classifier (LGBMClassifier): A gradient boosting framework that grows trees leaf-wise, designed for efficiency and scalability in large datasets [9].
- AdaBoost Classifier (AdaBoostClassifier): A boosting algorithm that focuses on correcting the errors of previous classifiers, increasing model accuracy [10].
- Bernoulli Naive Bayes (BernoulliNB): A simple and fast classifier based on Bayes' theorem, particularly suited for binary features [11].
- Support Vector Classifier (SVC): A powerful classifier that finds a hyperplane to separate classes, known for effectiveness in high-dimensional spaces [12,13].

By evaluating these classifiers using Lazy Predict, we gained insights into their performance, leading us to select the most effective models for sexism classification based on F1 score and accuracy.

2. Exist 2024 Contest and Task 1 Overview

2.1 Exist 2024

The EXIST 2024 competition centers around the identification of sexism on social media, particularly within tweets. The main task of the competition is a binary classification problem, where systems must determine whether a tweet contains sexist expressions or behaviors. This includes tweets that are sexist, describe a sexist situation, or critique sexist conduct.

For example, the following tweets demonstrate instances of both sexist and non-sexist content:
Sexist:

- “Mujer al volante, tenga cuidado!”
- “People really try to convince women with little to no ass that they should go out and buy a body.
- Like bih, I don’t need a fat ass to get a man. Never have.”
- Not Sexist:
- “Alguien me explica que zorra hace la gente en el cajero que se demora tanto.”
- "@messyworldorder it’s honestly so embarrassing to watch and they’ll be like ’not all white women are like that.’"

2.2 Task 1

In Task 1, participants are required to create models that classify tweets into these two categories. The challenge lies in distinguishing the nuanced language and context that reveal sexism. The objective is to design models capable of identifying not only overtly sexist comments but also more subtle and context-dependent expressions of sexism.

The development and evaluation process for these models involves multiple stages, including data preprocessing, feature extraction, and the use of diverse machine learning algorithms. The ultimate goal is to develop reliable tools that contribute to reducing sexism on social media platforms, thus fostering a more respectful online discourse.

3. Sexism Identification Methodology

Our methodology for identifying sexism in social media posts followed a systematic approach, using exclusively the training and development datasets. The primary goal was to train multiple machine learning models on the training dataset, then select the top-performing ones based on accuracy and F1 score on the development dataset, as specified by the competition. Our approach builds on previous work that addressed similar sentiment classification tasks utilizing comparisons of different embedding methods and regression classifiers [14,15].

3.1 Text Embedding

We began by applying text embedding techniques to represent the textual data in vectorized form. Specifically, we used Term Frequency-Inverse Document Frequency (TF-IDF) for each language in our dataset. TF-IDF transforms text into numerical vectors, based on the frequency of terms in individual documents compared to the entire document set. Our experiments involved several configurations, including:

- Different feature types, such as words, characters, and character n-grams (e.g., bigrams, trigrams).
- Various feature ranges, from single words to sequences of characters of varying lengths.
- Different feature amounts, ranging from 1,000 to 20,000, to determine the optimal number of features for classification.

3.2 Text Preprocessing

Text preprocessing plays a crucial role in Natural Language Processing, especially for tasks like sexism identification. Social media texts often contain various types of noise, such as typos, emojis, slang, HTML tags, spelling errors, and repeated letters. If not properly handled, this noise can significantly impact model performance and lead to incorrect analyses.

Previous studies have explored the effects of combining multiple preprocessing methods on text classification across different datasets [16,17]. Their findings emphasize the importance of applying diverse preprocessing techniques systematically. Combining these techniques with machine learning can considerably improve classification accuracy.

In our work, we implemented a thorough preprocessing strategy to clean and standardize the textual data before further analysis. This ensured that the models received high-quality inputs, enhancing their ability to correctly identify and classify sexist content.

3.3 Lazy Predict

Lazy Predict is an open-source Python library that simplifies the process of building and comparing multiple machine learning models. It allows for the quick benchmarking of different algorithms without extensive manual coding, providing a streamlined interface for efficient model evaluation [18].

In the context of sexism identification, Lazy Predict was particularly useful during the initial model selection phase. With a wide variety of machine learning classifiers available, we needed a systematic way to compare their performance on our dataset. Lazy Predict automatically trained and evaluated multiple models with default hyperparameters, giving us a comprehensive overview of the algorithms most suitable for our task.

Lazy Predict compared a range of machine learning classifiers, including: Ada Boost Classifier, Bagging Classifier, Bernoulli NB, Calibrated Classifier CV, Decision Tree Classifier, Dummy Classifier, Extra Tree Classifier, ExtraTreesClassifier, Gaussian NB, KNeighbors Classifier, NuSVC, Passive Aggressive Classifier, Perceptron, Quadratic Discriminant Analysis, Random Forest Classifier, Ridge Classifier, Ridge Classifier CV, SGD Classifier, SVC, and LGBM Classifier. The results of these comparisons on our data are presented in Table 1. (Appendices).

Model	**Accuracy**	**Balanced Accuracy**	**F1 Score**	**Time Taken**
ExtraTreesClassifier	0.734104046	0.731715653	0.731389883	82.78278661
LGBM Classifier	0.726396917	0.724293441	0.724220837	6.884508371
Random Forest Classifier	0.716763006	0.713832506	0.712489727	29.08332467
Bagging Classifier	0.706165703	0.703156112	0.701514649	157.2697315
AdaBoost Classifier	0.695568401	0.692479717	0.690518216	51.01095295
Bernoulli NB	0.691714836	0.690796903	0.691232741	2.754544497
SVC	0.685934489	0.682405123	0.679168563	233.5694647
NuSVC	0.681117534	0.678913192	0.678410111	232.2601142
Nearest Centroid	0.675337187	0.674771167	0.675151581	2.109311104
Decision Tree Classifier	0.671483622	0.670094208	0.670357222	33.56897783
Perceptron	0.661849711	0.660454248	0.660690278	4.56968379
Extra Tree Classifier	0.654142582	0.653128622	0.653515573	2.80695343
SGD Classifier	0.655105973	0.651847009	0.648841001	5.900460243
Logistic Regression	0.650289017	0.648934589	0.649169867	8.320355654
Passive Aggressive Classifier	0.647398844	0.646404797	0.646789552	7.302331448
Linear SVC	0.628131021	0.62717317	0.627549493	69.49884391
Linear Discriminant Analysis	0.619460501	0.619342328	0.619497601	159.9661644
Ridge Classifier	0.619460501	0.619342328	0.619497601	11.12075329
Calibrated Classifier CV	0.625240848	0.619234598	0.601675594	294.2782121
Ridge Classifier CV	0.61849711	0.618402479	0.618539615	162.7189815

Table 1: Lazy Predict Results

3.4 Model Training and Selection

Once the text data was vectorized, we proceeded with training a diverse range of machine learning models on the training dataset. These models included:

- Extra Trees Classifier
- LightGBM Classifier
- Random Forest Classifier
- AdaBoost Classifier
- Bernoulli Naive Bayes
- Support Vector Classifier (SVC)

For each model, we evaluated its accuracy and F1 score on the development dataset. We experimented with different feature combinations to fine-tune performance. The models showing the best results on the development dataset were selected for further evaluation.

3.5 Test Prediction

Finally, to decide which models would label the test dataset, we formed three groups of models: the top 10, top 50, and top 100 models. Each group was used to label the test dataset. For each tweet, we selected the majority label (sexist or not sexist) and generated a JSON file containing the predictions.

4. Results

Table presents the accuracy rankings and F1 scores of the models for Task 1. For each language, the table shows the top-performing model, feature type, feature range, number of features, preprocessing details, classifier type, and the corresponding scores from the development phase.

Classifier	Type	Range	Amount	Preprocessing	Accuracy	F1
ExtraTreesClassifier	char	6	20000	remove_punctuation	0.7649	0.7640
ExtraTreesClassifier	char	6	10000	remove_spaces	0.7649	0.7640
RandomForestClassifier	char	6	10000	remove_punctuation	0.7649	0.7631
RandomForestClassifier	char	6	17500	remove_punctuation	0.7620	0.7600
ExtraTreesClassifier	char	6	10000	remove_punctuation	0.7611	0.7600
RandomForestClassifier	char	6	15000	None	0.7592	0.7567
ExtraTreesClassifier	char	6	17500	None	0.7582	0.7573
ExtraTreesClassifier	char	6	7500	remove_numerical_punct_spaces	0.7582	0.7572
ExtraTreesClassifier	char	6	12500	remove_spaces	0.7582	0.7572
ExtraTreesClassifier	char	6	7500	remove_spaces	0.7572	0.7562
ExtraTreesClassifier	char	6	7500	remove_punctuation	0.7572	0.7562
ExtraTreesClassifier	char	6	12500	remove_punctuation	0.7563	0.7553
ExtraTreesClassifier	char	6	15000	None	0.7563	0.7551
LGBMClassifier	char	3	17500	None	0.7563	0.7537
LGBMClassifier	char	3	17500	remove_punctuation	0.7563	0.7537
LGBMClassifier	char	3	17500	remove_spaces	0.7563	0.7537
LGBMClassifier	char	3	17500	remove_numerical_punct_spaces	0.7563	0.7537
ExtraTreesClassifier	char	6	10000	remove_numerical_punct_spaces	0.7553	0.7545
RandomForestClassifier	char	6	15000	remove_numerical_punct_spaces	0.7553	0.7527
ExtraTreesClassifier	char	6	10000	None	0.7543	0.7534
RandomForestClassifier	char	6	12500	remove_punctuation	0.7543	0.7526
LGBMClassifier	char_wb	3	17500	None	0.7543	0.7522
LGBMClassifier	char_wb	3	17500	remove_punctuation	0.7543	0.7522
LGBMClassifier	char_wb	3	17500	remove_spaces	0.7543	0.7522
LGBMClassifier	char_wb	3	17500	remove_numerical_punct_spaces	0.7543	0.7522
RandomForestClassifier	char	6	17500	None	0.7543	0.7520
RandomForestClassifier	char	6	17500	remove_spaces	0.7543	0.7519
RandomForestClassifier	char	6	12500	None	0.7534	0.7515
LGBMClassifier	char	3	15000	None	0.7534	0.7512
LGBMClassifier	char	3	15000	remove_punctuation	0.7534	0.7512
LGBMClassifier	char	3	15000	remove_spaces	0.7534	0.7512
LGBMClassifier	char	3	15000	remove_numerical_punct_spaces	0.7534	0.7512
LGBMClassifier	char	3	12500	None	0.7534	0.7511
LGBMClassifier	char	3	12500	remove_punctuation	0.7534	0.7511
LGBMClassifier	char	3	12500	remove_spaces	0.7534	0.7511
LGBMClassifier	char	3	12500	remove_numerical_punct_spaces	0.7534	0.7511
RandomForestClassifier	char	6	20000	remove_spaces	0.7534	0.7509
ExtraTreesClassifier	char	6	15000	remove_numerical_punct_spaces	0.7524	0.7516
ExtraTreesClassifier	char	6	17500	remove_spaces	0.7524	0.7516
ExtraTreesClassifier	char_wb	6	5000	remove_spaces	0.7524	0.7507
RandomForestClassifier	char	6	15000	remove_punctuation	0.7524	0.7505
RandomForestClassifier	char	6	20000	remove_punctuation	0.7524	0.7500
LGBMClassifier	char_wb	3	2500	None	0.7524	0.7494
LGBMClassifier	char_wb	3	2500	remove_punctuation	0.7524	0.7494
LGBMClassifier	char_wb	3	2500	remove_spaces	0.7524	0.7494

LGBMClassifier	char_wb	3	2500	remove_numerical_punct_spaces	0.7524	0.7494
ExtraTreesClassifier	char	6	20000	remove_numerical_punct_spaces	0.7514	0.7504
LGBMClassifier	char_wb	3	5000	None	0.7514	0.7496
LGBMClassifier	char_wb	3	5000	remove_punctuation	0.7514	0.7496
LGBMClassifier	char_wb	3	5000	remove_spaces	0.7514	0.7496

Table 2: 50 Best Results

The most prominent classifiers among the top models were the ExtraTreesClassifier, Random Forest Classifier, and LGBM Classifier, all of which are based on ensemble learning techniques such as random forests and boosting. Naive Bayes, a simpler yet effective classifier, also performed well.

While preprocessing is often considered beneficial in machine learning tasks, there was a balance between models that used preprocessing and those that worked better with raw text. More advanced preprocessing methods, such as stemming or lemmatization, might yield further improvements.

In terms of feature types, character sequences outperformed word sequences, with medium-length character n-grams (approximately six characters) proving more effective than shorter or longer sequences. Additionally, models with over 10,000 features generally performed better, indicating the need for a rich feature set to capture subtle nuances in the tweets.

Our best submission, a combination of the top 50 models, ranked 39th in the competition. A second submission, combining the top 100 models, ranked 41st, while the submission using only the top 10 models ranked 47th.

5. Conclusions

This paper outlines our participation in the EXIST 2024 competition, focusing on the task of sexism identification in tweets. Through extensive experimentation with models, text preprocessing methods, feature types, and feature amounts, we identified the most effective models based on accuracy and F1 score on the development dataset.

The ExtraTreesClassifier, Random Forest Classifier, and LGBM Classifier emerged as the top-performing models, leveraging ensemble learning techniques like bagging and boosting. We observed a balance between models with and without preprocessing, suggesting that while preprocessing can improve performance, it is not universally necessary. Character n-grams, particularly medium-length sequences, proved more effective than word sequences for capturing sexist language, and a larger feature set generally led to better results.

Overall, our study highlights the complexity of sexism identification on social media and the importance of leveraging a variety of techniques and models for robust performance. These insights contribute to ongoing efforts to build accurate and reliable models for sexism detection on online platforms.

Future Work

This work opens several directions for future research and improvement. A significant avenue is exploring advanced preprocessing methods like stemming, lemmatization, and context-aware normalization. These techniques could improve model generalization and robustness in handling linguistic variation.

Augmenting the training dataset with more examples from diverse sources and languages is another important step, as it could enhance the models' ability to generalize across different contexts and cultures.

Conducting detailed error analysis to understand recurring misclassifications, such as sarcasm, irony, or cultural references, is also critical. This analysis can help improve model accuracy.

Exploring additional feature types, including domain-specific features, may provide a more nuanced understanding of sexist language. Incorporating semantic and syntactic features or external knowledge bases could also yield improvements.

Finally, extending this research to include deep learning models like BERT or Transformers for sexism identification is a promising direction, particularly in handling the unique challenges of various languages with different morphological structures and idiomatic expressions.

By addressing these areas, we aim to further improve the effectiveness of sexism identification models, contributing to the broader goal of reducing sexism and promoting equality.

References

1. Jha, A., Mamidi, R. (2017). When does a compliment become sexist? analysis and classification of ambivalent sexism using twitter data, in: Proceedings of the Second Workshop on NLP and Computational Social Science.
2. Rodríguez-Sánchez, F., Carrillo-de-Albornoz, J., & Plaza, L. (2020). Automatic classification of sexism in social networks: An empirical study on twitter data. IEEE Access, 8, 219563-219576.
3. Plaza, L., Carrillo-de-Albornoz, J., Morante, R., Amigó, E., Gonzalo, J., Spina, D., & Rosso, P. (2023, September). Overview of exist 2023–learning with disagreement for sexism identification and characterization. In International Conference of the Cross-Language Evaluation Forum for European Languages (pp. 316-342). Cham: Springer Nature Switzerland.
4. Plaza, L., Carrillo-de-Albornoz, J., Morante, R., Amigó,

E., Gonzalo, J., Spina, D., & Rosso, P. (2023, September). Overview of exist 2023–learning with disagreement for sexism identification and characterization. In International Conference of the Cross-Language Evaluation Forum for European Languages (pp. 316-342). Cham: Springer Nature Switzerland.

5. Ramos, J. (2003). Using tf-idf to determine word relevance in document queries. In Proceedings of the first instructional conference on machine learning (Vol. 242, No. 1, pp. 29-48).
6. Breiman, L. (1996). Bagging predictors. Machine learning, 24, 123-140.
7. Breiman, L. (2001). Random forests. Machine learning, 45, 5-32.
8. Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. Machine learning, 63, 3-42.
9. Alzamzami, F., Hoda, M., & El Saddik, A. (2020). Light gradient boosting machine for general sentiment classification on short texts: a comparative evaluation. IEEE access, 8, 101840-101858.
10. Schapire, R. E. (2013). Explaining adaboost. In Empirical inference: festschrift in honor of vladimir N. Vapnik (pp. 37-52). Berlin, Heidelberg: Springer Berlin Heidelberg.
11. Kim, S. B., Han, K. S., Rim, H. C., & Myaeng, S. H. (2006). Some effective techniques for naive bayes text classification. IEEE transactions on knowledge and data engineering, 18(11), 1457-1466.
12. Vapnik, V. (1995). Support-vector networks. Machine learning, 20, 273-297.
13. Chang, C. C., & Lin, C. J. (2011). LIBSVM: a library for support vector machines. ACM transactions on intelligent systems and technology (TIST), 2(3), 1-27.
14. Keinan, R., & HaCohen-Kerner, Y. (2023, July). JCT at SemEval-2023 Tasks 12 A and 12B: Sentiment Analysis for Tweets Written in Low-resource African Languages using Various Machine Learning and Deep Learning Methods, Resampling, and HyperParameter Tuning. In Proceedings of the 17th International Workshop on Semantic Evaluation (SemEval-2023) (pp. 365-378).
15. Keinan, R. (2024). Text Mining at SemEval-2024 Task 1: Evaluating Semantic Textual Relatedness in Low-resource Languages using Various Embedding Methods and Machine Learning Regression Models. In Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024) (pp. 420-431).
16. HaCohen-Kerner, Y., Yigal, Y., & Miller, D. (2019). The Impact of Preprocessing on the Classification of Mental Disorders. In ICDM (Posters) (pp. 52-66).
17. HaCohen-Kerner, Y., Miller, D., & Yigal, Y. (2020). The influence of preprocessing on text classification using a bag-of-words representation. PloS one, 15(5), e0232525.
18. Putra, M. I. J., & Alexander, V. (2023). Comparison of Machine Learning Land Use-Land Cover Supervised Classifiers Performance on Satellite Imagery Sentinel 2 using Lazy Predict Library. Indonesian Journal of Data and Science, 4(3), 183-189.