# Sexism Identification in Social Networks using TF-IDF Embeddings, PreProccessing, Feature Selection, Word/Char N-Grams and Various Machine Learning Models In Spanish and English

Ron Keinan[1,*,†]

[1]*Department of Computer Science, Jerusalem College of Technology, Lev Academic Center.*

**Abstract**

In this paper, we describe our submission to the EXIST-2024 contest. We tackled Task 1 - "Sexism Identification in tweets" in English and Spanish. To classify the tweets as texts containing sexism, we created different set up of models, changing the ML classifier, the feature type(word/char), the feature amount and the preprocessing of the text. With this set up, we vectorized the text data using tf-idf embedding technique. After training all these set-ups on the training dataset, we chose the best models according to their accuracy and F1-score on the dev set, and used them to predict the test labels. The best model got a F1 score of 72.23 and the rank of 39 out of 70.

**Keywords**

Sexism identification, machine learning, TF-IDF, feature selection, char based n-grams,

## 1. Introduction

Sexism identification in social networks has emerged as a significant challenge within the field of Natural Language Processing. This task involves detecting and classifying sexist content within social media posts, which is crucial for maintaining respectful and inclusive online environments. The identification of sexist remarks is not only important for individual platforms to manage content but also for broader societal implications, such as monitoring and mitigating the spread of harmful stereotypes and promoting gender equality[1].

Social networks have become the primary platforms for social complaints, activism, and widespread movements such as MeToo, 8M, and Time'sUp. These movements have gained momentum quickly, with countless women around the world sharing their experiences of abuse, discrimination, and other forms of sexism encountered in their daily lives. While social networks play a crucial role in amplifying voices against injustice, they also serve as conduits for the transmission of sexism and other disrespectful and hateful behaviors[2].

In this context, the development of automatic tools for sexism identification is essential. These tools can aid in detecting and flagging sexist behaviors, providing real-time alerts to help manage and moderate online content. Furthermore, they enable the estimation of the prevalence of sexist and abusive situations on social media platforms. By analyzing patterns and forms of sexism, these tools can offer insights into how sexism is expressed and propagated in these digital spaces.

The significance of this task lies in its potential to enhance the safety and inclusivity of social media environments. Effective sexism identification tools can not only assist in immediate content moderation but also contribute to long-term strategies for reducing the spread of harmful stereotypes and fostering a more respectful online community. The efforts in this area, including the contributions from this lab, are pivotal in developing robust applications aimed at detecting and mitigating sexism in social networks.

*CLEF 2024: Conference and Labs of the Evaluation Forum, September 09–12, 2024, Grenoble, France*

*Corresponding author.

†These authors contributed equally.

ronke21@gmail.com (R. Keinan)

0009-0006-3122-6143 (R. Keinan)

In this paper, we describe our participation in the EXIST-2024 contest[3][4], specifically addressing Task 1 - "Sexism Identification in tweets" in English and Spanish. The approach to solving this task involved creating multiple models by varying several key components: the machine learning classifier used, the type of features (word-level or character-level), the number of features, and the preprocessing techniques applied to the text data. Subsequently, we vectorized the text data using the Term Frequency-Inverse Document Frequency (TF-IDF) embedding technique.

The importance of this task is underscored by the growing volume of user-generated content on social media platforms, where the rapid identification and mitigation of sexist content can significantly impact user experience and safety. By leveraging a combination of preprocessing, feature selection, and various machine learning models, our approach contributes to the ongoing efforts in developing robust automated systems for sexism detection.

## 2. Theoretical Review

### 2.1. Feature Selection

Feature selection is a critical process in text classification tasks, significantly impacting model performance by identifying the most informative attributes from the text data. In our approach to sexism identification, we meticulously focused on selecting features based on two primary types: word n-grams and character n-grams.

#### 2.1.1. Word N-grams

Word n-grams represent contiguous sequences of words within the text, capturing contextual relationships and syntactic structures. By considering sequences of words, n-grams facilitate the model's understanding of semantic meaning conveyed through word combinations. For instance, in a bigram model, pairs of consecutive words are considered, while a trigram model examines sequences of three words. Using the sentence "The quick brown fox jumps over the lazy dog" as an example, the bigrams include "The quick", "quick brown", "brown fox", "fox jumps", "jumps over", "over the", "the lazy", and "lazy dog". Trigrams, on the other hand, include sequences like "The quick brown", "quick brown fox", "brown fox jumps", "fox jumps over", "jumps over the", "over the lazy", and "the lazy dog". This granular approach helps in capturing the syntactic structure and semantic nuances of word combinations, which are pivotal for understanding context-dependent expressions of sexism.

However, word n-grams have limitations, particularly when dealing with sparse data and out-of-vocabulary words, which are prevalent in social media texts. To mitigate these issues, we implemented techniques such as TF-IDF weighting to emphasize the importance of rare but informative n-grams and reduce the impact of common but less informative ones.

#### 2.1.2. Character N-grams

Character n-grams, especially those with word boundaries (char-wb), segment the text into sequences of characters while respecting word boundaries. This method is adept at capturing morphological patterns and handling variations such as typos, slang, and informal language, which are ubiquitous in social media. For instance, character n-grams of length six in the word "identification" might include "identi", "dentif", "entifi", and so on. By incorporating word boundaries, char-wb n-grams can maintain the integrity of individual words while allowing the model to learn from character-level patterns.

Our experiments demonstrated that character n-grams, particularly of medium length (around six characters), consistently outperformed word n-grams. This indicates their superior ability to capture the nuanced morphological features and informal linguistic variations typical in sexist language. The flexibility of character n-grams in handling different morphological structures and idiomatic expressions was particularly beneficial in our dataset, which included diverse and colloquial expressions of sexism.

### 2.1.3. Comparative Analysis

Through extensive experimentation, we observed that models utilizing character n-grams with word boundaries achieved higher accuracy and F1 scores compared to those relying solely on word n-grams. This suggests that character n-grams provide a richer and more robust feature set for sexism identification in tweets, capable of capturing subtle and context-dependent expressions of sexism that may be overlooked by word n-grams alone.

### 2.1.4. TF-IDF Embeddings

To optimize the feature selection process, we employed the Term Frequency-Inverse Document Frequency (TF-IDF) technique. TF-IDF helps in quantifying the importance of each n-gram by balancing its frequency within a document against its frequency across all documents in the dataset. By doing so, it highlights the most informative features that are likely to contribute to the classification task.

## 2.2. Text Embeddings

Text embeddings are representations of textual data in a continuous vector space, enabling algorithms to process and analyze text effectively. These embeddings capture both semantic and syntactic similarities between words or documents, facilitating various Natural Language Processing (NLP) tasks such as sentiment analysis, document classification, and information retrieval.

### 2.2.1. Types of Text Embeddings

There are several types of text embeddings, each with its unique characteristics and applications:

**Word Embeddings** Word embeddings, such as Word2Vec and GloVe, map each word to a high-dimensional vector, capturing semantic relationships based on the context in which words appear. For instance, words with similar meanings (e.g., "king" and "queen") are located close to each other in the vector space, while unrelated words are far apart. Word embeddings are particularly useful for tasks that require understanding word semantics, such as word analogy tasks and semantic similarity.

**Contextualized Word Embeddings** Contextualized word embeddings, such as those generated by models like ELMo, BERT, and GPT, provide representations that vary depending on the word's context in a sentence. Unlike static word embeddings, these embeddings can capture the polysemy of words (i.e., words with multiple meanings). For example, the word "bank" will have different embeddings in the sentences "I sat on the bank of the river" and "I deposited money in the bank." This context-awareness significantly improves performance in tasks like named entity recognition, question answering, and machine translation.

**Document Embeddings** Document embeddings extend the concept of word embeddings to larger text units, such as sentences, paragraphs, or entire documents. Techniques like Doc2Vec and Universal Sentence Encoder create fixed-length vectors that represent the overall meaning of a text segment. These embeddings are valuable for tasks such as document classification, clustering, and information retrieval, where the goal is to compare and analyze entire documents rather than individual words.

### 2.2.2. Significance in NLP

The use of text embeddings represents a significant advancement in NLP, as they provide a dense and continuous representation of text that traditional bag-of-words models cannot achieve. Embeddings allow for the efficient handling of large vocabularies and capture intricate relationships between words and phrases. This has led to substantial improvements in various NLP tasks, making embeddings a crucial component of modern NLP systems.

#### 2.2.3. TF-IDF Embeddings

In our study, we utilized Term Frequency-Inverse Document Frequency (TF-IDF), as an embedding method. [5]

TF-IDF (Term Frequency-Inverse Document Frequency) is a statistical measure used to evaluate the importance of a word in a document relative to a collection of documents. It calculates a weight for each word based on its frequency in the document and its inverse frequency across all documents. Words with high TF-IDF scores are considered more informative for distinguishing documents (Ramos, 2003).

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

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

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.

### 2.3. Machine Learning Classifiers

In the approach to sexism identification, we experimented with a variety of machine learning classifiers to determine the most effective model for our task. Each classifier brings unique strengths and characteristics, making them suitable for different aspects of the classification problem. The classifiers where chosen from highest accuray models from Lazy Predict. Below, we describe the key classifiers we employed:

1. **Random Forest Classifier (`RandomForestClassifier`):**
   - The Random Forest Classifier is another ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes for classification tasks[6][7]. By averaging the results from multiple trees, it enhances predictive accuracy and controls over fitting. Random forests are particularly effective for datasets with a large number of features and complex, non-linear relationships.
2. **Extra Trees Classifier (`ExtraTreesClassifier`):**
   - The Extra Trees Classifier is an ensemble learning method that aggregates the results of multiple unpruned decision trees, generated from random subsets of the training data and features[8]. This technique enhances the model's robustness and accuracy by reducing variance and preventing overfitting. It is known for its high performance in handling large datasets and capturing complex interactions among features.
3. **LightGBM Classifier (`LGBMClassifier`):**
   - LightGBM is a gradient boosting framework that uses tree-based learning algorithms. It is designed to be efficient and scalable, making it Ill-suited for large datasets and high-dimensional data[9]. LightGBM incorporates techniques such as leaf-wise tree growth and histogram-based decision tree learning, which improve speed and accuracy while maintaining low memory usage. It excels in handling categorical features and complex data structures.
4. **AdaBoost Classifier (`AdaBoostClassifier`):**

- AdaBoost, short for Adaptive Boosting, combines the predictions of several 'ak classifiers to create a strong classifier[10]. It works by sequentially training classifiers, each focusing on the errors made by the previous ones. This iterative approach allows AdaBoost to improve the model's performance by emphasizing the difficult-to-classify instances. It is versatile and can be used with various base learners, though it is most commonly paired with decision trees.

5. **Bernoulli Naive Bayes (`BernoulliNB`):**
   - The Bernoulli Naive Bayes classifier is based on Bayes' theorem and assumes that features follow a Bernoulli distribution (binary or boolean values). It is especially suited for binary/boolean features and is effective for tasks where the presence or absence of a feature is more important than its frequency. This classifier is simple, fast, and performs Ill on high-dimensional sparse datasets.[11]
6. **Support Vector Classifier (`SVC`):**
   - The Support Vector Classifier is a powerful and versatile classifier that constructs a hyperplane or set of hyperplanes in a high-dimensional space to separate different classes. It is particularly effective in high-dimensional spaces and for cases where the number of dimensions exceeds the number of samples. SVC is robust to overfitting, especially in high-dimensional space, and can be extended to handle non-linear classification using kernel functions[12][13].

By evaluating these classifiers using LazyPredict, we was able to quickly identify which models performed best on our dataset. This informed our decision-making process and guided us in selecting and fine-tuning the models that ultimately provided the highest accuracy and F1 scores for sexism identification in social networks.

## 3. EXIST 2024 Contest and Task 1 Overview

### 3.1. EXIST 2024

The EXIST 2024 competition[3] focuses on the identification of sexism in social media, with a particular emphasis on analyzing tweets. The primary task within this competition is a binary classification problem, where systems must determine whether a given tweet contains sexist expressions or behaviors. This includes tweets that are sexist themselves, describe a sexist situation, or criticize sexist behavior.

For instance, the following tweets illustrate examples of sexist and non-sexist messages:

**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'"

### 3.2. Task 1

In Task 1, participants are required to develop models that can accurately classify tweets into these two categories. The challenge lies in creating a system that can effectively discern the subtle nuances of language and context that indicate sexism. The objective is to build models that are not only precise in detecting overtly sexist remarks but also adept at identifying more covert and context-dependent expressions of sexism.

The development and evaluation of these models involve several stages, including data preprocessing, feature extraction, and the application of various machine learning algorithms. The ultimate goal is to create robust and reliable tools that can contribute to the broader effort of mitigating sexism on social media platforms, thereby promoting a healthier and more respectful online discourse.

## 4. Sexism Identification Methodology

Our methodology for identifying sexism in social media posts was based on a systematic approach using training and development datasets exclusively. The primary objective was to train various machine learning models on the training dataset and then select the best-performing models based on their accuracy and F1 score, as stipulated by the competition requirements, on the development dataset. our approach to solving the task was based on a previous study that dealt with a similar sentiment classification task [14][15] and was based on a comparison of different embedding methods and then a comparison between different regression classifiers.

### 4.1. Text Embedding

we began by employing text embedding techniques to represent the textual data in a vectorized format. Specifically, we utilized the Term Frequency-Inverse Document Frequency (TF-IDF) method for each language in our dataset. TF-IDF transforms text into numerical vectors based on the frequency of terms within documents relative to a collection of documents. we experimented with different configurations, including:

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

### 4.2. Text PreProcessing

Text preprocessing is a critical step in Natural Language Processing, especially in tasks such as Sexism Identification. In both general and social media text documents, various types of noise are commonly present. This noise can include typos, emojis, slang, HTML tags, spelling mistakes, and repetitive letters. If the text is not properly preprocessed, it can lead to incorrect analysis outcomes and significantly impact the performance of the models.

Former researchers[16][17] explored the effects of all possible combinations of six preprocessing methods on text classification across three different datasets. Their main conclusion emphasized the importance of systematically applying a variety of preprocessing techniques. By combining these preprocessing methods with multiple machine learning approaches, the accuracy of text classification can be substantially improved.

In our work, we adopted a comprehensive preprocessing strategy to clean and standardize the text data before applying further analytical techniques. This approach ensured that the models received high-quality input, thereby enhancing their ability to accurately identify and classify sexist content in social media posts.

### 4.3. Lazy Predict

LazyPredict is an open-source Python library designed to streamline the process of building and comparing multiple machine learning models. It is particularly useful for quickly benchmarking different algorithms without the need for extensive manual coding. By providing a simple interface, LazyPredict allows data scientists to efficiently identify the most promising models for their specific tasks[18].

In the context of sexism identification task, LazyPredict proved to be a valuable tool during the initial model selection phase. Given the variety of machine learning classifiers available, we needed a systematic way to evaluate their performance on the dataset. LazyPredict facilitated this by automatically training and testing a wide array of models using default hyper parameters, enabling us to gain a broad understanding of which algorithms might be most effective for our problem.

LazyPredict compared the following ML classifiers: AdaBoostClassifier, BaggingClassifier, BernoulliNB, CalibratedClassifierCV, DecisionTreeClassifier, DummyClassifier, ExtraTreeClassifier, ExtraTreesClassifier, GaussianNB, KNeighborsClassifier, NuSVC, PassiveAggressiveClassifier, Perceptron, QuadraticDiscriminantAnalysis, RandomForestClassifier, RidgeClassifier, RidgeClassifierCV, SGDClassifier, SVC, LGBMClassifier. The results of the LazyPredict model on the data is presented in Table 1 (Appendices).

### 4.4. Model Training and Selection

With the vectorized representations of the text data, we proceeded to train multiple machine learning models using the training dataset. we explored a diverse range of classifiers, including but not limited to:

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

For each model, we evaluated its performance on the development dataset based on accuracy and F1 score. we experimented with different feature combinations to optimize model performance. The models that demonstrated the highest accuracy and F1 score on the development dataset Ire selected as our best-performing models for further evaluation.

### 4.5. Test Prediction

Finally, we got a list of all the best models. To choose the models that will label the test pool and the labels that will be accessible to the competition, we chose 3 groups of models. The 10 best models, the 50 best models, and the 100 best models. we asked each group of models to tag the test database, for each tweet we chose the majority of tags (yes or no) and created a JSON file that contains all the answers.

## 5. Results

Table 2 (Appendices) presents the Accuracy rank and F1 score of the models for Task 1. The table shows for each language the ideal model we received, feature type, range and amount, whether it performed pre-processing, which classifier it used, what was the score we received in the dev phase.

The most prominent classifiers in the best models are the ExtraTreesClassifier, RandomForestClassifier, LGBMClassifier. They are based on classic machine learning algorithms - Random Forest and boosting, and Naive Bayes which are recognized as classic classifiers but strong and good in many ML tasks. Despite the well-known advantages of preprocessing methods in ML tasks, it seems that there is an overall balance between models that were quicker to preprocess their text and models that worked better on the raw text. It may be that more advanced preprocessing methods such as stemming or lemmatization will be more helpful for learning.

With respect to the type of features, sequences of characters seem to work much better than sequences of words. And precisely a medium group of about 6 characters was better than low ranges of 3 or high ranges of 9.

Regarding the amount of features, it seems that more than 10000 features were often required to obtain the good models, and low amounts converged to lower accuracy.

The best model we sent was the combination of the results of the top 50 models and it came in 39th place in the competition. The second model was a combination of the 100 best models in the competition and it was ranked 41st. The model of the 10 best models was ranked 47th.

## 6. Conclusions

In this paper, we described our participation in the EXIST-2024 competition, focusing on the task of sexism identification in tweets. Our approach involved experimenting with various models, text preprocessing techniques, feature types, and feature amounts. Through systematic experimentation and evaluation, we identified the most effective models based on accuracy and F1 score on the development dataset.

Our findings revealed several key insights. First, the ExtraTreesClassifier, RandomForestClassifier, and LGBMClassifier emerged as the top-performing models. These classifiers, based on ensemble learning techniques such as bagging and boosting, demonstrated strong performance across various configurations. Additionally, we observed a balance between models that utilized text preprocessing and those that did not. While preprocessing methods like stemming and lemmatization can potentially enhance model performance by normalizing text, their impact varied, suggesting the need for more advanced and context-specific preprocessing techniques.

Moreover, character sequences generally outperformed word sequences, with character n-grams of medium length (around six characters) providing better results compared to shorter or longer sequences. This finding highlights the effectiveness of character n-grams in capturing the nuances of sexist language. Furthermore, models with more than 10,000 features tended to perform better, underscoring the importance of a rich feature set for capturing the subtleties in tweets.

Overall, our study underscores the complexity of sexism identification in social media posts and the importance of leveraging diverse techniques and models to achieve robust performance. These insights contribute to the ongoing development of more accurate and reliable models for sexism detection in online platforms.

## 7. Future Work

Our current work opens several avenues for future research and improvements. One significant direction is the investigation of advanced preprocessing techniques, such as stemming, lemmatization, and context-aware normalization. These sophisticated methods could enhance the robustness and generalization of our models by better handling linguistic variations and subtleties.

Additionally, enriching the training dataset with more examples from diverse sources and languages is essential. This augmentation could improve the models' ability to generalize across different contexts and cultural nuances, thereby enhancing their performance.

Conducting in-depth error analysis is another crucial area for future work. By thoroughly analyzing recurrent misclassifications and patterns, we can understand the root causes of these errors, such as sarcasm, irony, and cultural references. This understanding can inform the development of more accurate and reliable models.

Exploring additional feature types and combinations is also recommended. This includes investigating domain-specific features that better capture the nuances of sexist language. Incorporating semantic and syntactic features, as well as external knowledge sources, could provide a more comprehensive understanding of the data.

Lastly, extending our research to include deep learning models, such as BERT and Transformers, for sexism identification is a promising direction. Addressing the unique challenges posed by different languages, such as varying morphological structures and idiomatic expressions, will be critical in this endeavor.

By addressing these future directions, we aim further to enhance the effectiveness and applicability of sexism identification models, contributing to the broader goal of combating sexism and promoting equality in online spaces.

## References

[1] A. Jha, R. Mamidi, 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, 2017.

[2] F. Rodríguez-Sánchez, J. C. de Albornoz, L. Plaza, Automatic classification of sexism in social networks: An empirical study on twitter data, IEEE Access 8 (2020) 219563–219576.

[3] L. Plaza, J. Carrillo-de-Albornoz, V. Ruiz, A. Maeso, B. Chulvi, P. Rosso, E. Amigó, J. Gonzalo, R. Morante, D. Spina, Overview of EXIST 2024 – Learning with Disagreement for Sexism Identification and Characterization in Social Networks and Memes, in: Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Fifteenth International Conference of the CLEF Association (CLEF 2024), 2024.

[4] L. Plaza, J. Carrillo-de-Albornoz, V. Ruiz, A. Maeso, B. Chulvi, P. Rosso, E. Amigó, J. Gonzalo, R. Morante, D. Spina, Overview of EXIST 2024 – Learning with Disagreement for Sexism Identification and Characterization in Social Networks and Memes (Extended Overview), in: G. Faggioli, N. Ferro, P. Galuščáková, A. G. S. de Herrera (Eds.), Working Notes of CLEF 2024 – Conference and Labs of the Evaluation Forum, 2024.

[5] J. Ramos, Using tf-idf to determine word relevance in document queries, in: Proceedings of the First Instructional Conference on Machine Learning, volume 242, 2003.

[6] L. Breiman, Bagging predictors, Machine Learning 24 (1996) 123–140.

[7] L. Breiman, Random forests, Machine Learning 45 (2001) 5–32.

[8] P. Geurts, D. Ernst, L. Wehenkel, Extremely randomized trees, Machine Learning 63 (2006) 3–42.

[9] F. Alzamzami, M. Hoda, A. E. Saddik, Light gradient boosting machine for general sentiment classification on short texts: A comparative evaluation, IEEE Access 8 (2020) 101840–101858.

[10] R. E. Schapire, Explaining adaboost, in: Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 37–52.

[11] S.-B. Kim, K.-S. Han, H.-C. Rim, S. H. Myaeng, Some effective techniques for naive bayes text classification, IEEE Transactions on Knowledge and Data Engineering 18 (2006) 1457–1466. doi:10.1109/TKDE.2006.180.

[12] C. Cortes, V. Vapnik, Support-vector networks, Machine Learning 20 (1995) 273–297.

[13] C.-C. Chang, C.-J. Lin, Libsvm: A library for support vector machines, ACM Transactions on Intelligent Systems and Technology (TIST) 2 (2011) 1–27.

[14] R. Keinan, Y. HaCohen-Kerner, Jct at semeval-2023 tasks 12a 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), 2023.

[15] R. Keinan, 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: A. K. Ojha, A. S. Doğruöz, H. Tayyar Madabushi, G. Da San Martino, S. Rosenthal, A. Rosá (Eds.), Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024), Association for Computational Linguistics, Mexico City, Mexico, 2024, pp. 420–431. URL: https://aclanthology.org/2024.semeval-1.65.

[16] Y. HaCohen-Kerner, Y. Yigal, D. Miller, The impact of preprocessing on classification of mental

disorders, in: Proceedings of the 19th Industrial Conference on Data Mining (ICDM 2019), New York, 2019.

[17] Y. HaCohen-Kerner, D. Miller, Y. Yigal, The influence of preprocessing on text classification using a bag-of-words representation, PLOS ONE 15 (2020) e0232525.

[18] M. I. J. Putra, V. Alexander, 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 (2023) 183–189.

## 8. Appendices- Result Tables

**Table 1**
LazyPredict Results

| Model | Accuracy | Balanced Accuracy | F1 Score | Time Taken |
|---|---|---|---|---|
| ExtraTreesClassifier | 0.734104046 | 0.731715653 | 0.731389883 | 82.78278661 |
| LGBMClassifier | 0.726396917 | 0.724293441 | 0.724220837 | 6.884508371 |
| RandomForestClassifier | 0.716763006 | 0.713832506 | 0.712489727 | 29.08332467 |
| BaggingClassifier | 0.706165703 | 0.703156112 | 0.701514649 | 157.2697315 |
| AdaBoostClassifier | 0.695568401 | 0.692479717 | 0.690518216 | 51.01095295 |
| BernoulliNB | 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 |
| NearestCentroid | 0.675337187 | 0.674771167 | 0.675151581 | 2.109311104 |
| DecisionTreeClassifier | 0.671483622 | 0.670094208 | 0.670357222 | 33.56897783 |
| Perceptron | 0.661849711 | 0.660454248 | 0.660690278 | 4.56968379 |
| ExtraTreeClassifier | 0.654142582 | 0.653128622 | 0.653515573 | 2.80695343 |
| SGDClassifier | 0.655105973 | 0.651847009 | 0.648841001 | 5.900460243 |
| LogisticRegression | 0.650289017 | 0.648934589 | 0.649169867 | 8.320355654 |
| PassiveAggressiveClassifier | 0.647398844 | 0.646404797 | 0.646789552 | 7.302331448 |
| LinearSVC | 0.628131021 | 0.62717317 | 0.627549493 | 69.49884391 |
| LinearDiscriminantAnalysis | 0.619460501 | 0.619342328 | 0.619497601 | 159.9661644 |
| RidgeClassifier | 0.619460501 | 0.619342328 | 0.619497601 | 11.12075329 |
| CalibratedClassifierCV | 0.625240848 | 0.619234598 | 0.601675594 | 294.2782121 |
| RidgeClassifierCV | 0.61849711 | 0.618402479 | 0.618539615 | 162.7189815 |
| GaussianNB | 0.594412331 | 0.599300871 | 0.578915593 | 2.786282539 |
| QuadraticDiscriminantAnalysis | 0.544315992 | 0.55240869 | 0.492225354 | 131.178328 |
| KNeighborsClassifier | 0.523121387 | 0.511449077 | 0.388696557 | 4.114360094 |
| LabelSpreading | 0.514450867 | 0.501976285 | 0.351622593 | 15.14039254 |
| LabelPropagation | 0.514450867 | 0.501976285 | 0.351622593 | 14.17368817 |
| DummyClassifier | 0.512524085 | 0.5 | 0.347341163 | 1.893649578 |

**Table 2**
50 Best Results

| 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 |