Classification vs Clustering: an example on malignant comments

Text Mining & Search Project

Cervini Stella
Sabino Giuseppe

Contents

1 Introduction

The Internet, once praised for its open communication and limitless knowledge, has paradoxically turned into a breeding ground for malicious comments, a poisonous undercurrent threatening to contaminate the very essence of online discourse. These harmful remarks, from hateful speech to personal attacks, have cast a dark shadow over online spaces, eroding the safety and inclusivity that were once their defining features.

These malicious messages can have severe consequences, causing emotional distress, silencing marginalized voices, and even inciting real-world violence and their impact extends beyond individual experiences, permeating the broader online ecosystem. They create a chilling effect, discouraging open and honest conversations, especially among those already vulnerable to discrimination or oppression. This suppression of diverse perspectives hinders intellectual exchange and perpetuates harmful stereotypes.

In light of this alarming situation, the need for automated classification of malicious comments emerges as an essential requirement. This technology holds the potential to stem the tide of online abuse, creating a safer and more inclusive digital environment for all. By identifying and removing harmful content, automated classification systems can nurture an atmosphere of mutual respect and open dialogue.

However, developing and implementing effective automated classification systems is a complex endeavor. The diverse nature of malicious language, encompassing insults, threats, hate speech, and offensive humor, poses a significant challenge. Additionally, the inherent ambiguity and context-dependent nature of natural language further complicate the task of algorithmically distinguishing between malicious and benign comments.

2 Project goals

The project has the goal to provide an overview on the task of malignant comments detection by exploiting and comparing the results of two machine learning task: classification and clustering.
Classification assigns data points to predefined categories. This technique is often used for *supervised learning*, where the algorithm is trained on labeled data, meaning it's given examples of data points with their correct category labels. By learning from labeled data, the algorithm can identify the patterns that distinguish between the different categories. Once trained, the algorithm can be used to classify new data points, assigning them to the appropriate category.
Clustering, on the other hand, groups data points together based on their similarity. It's like organizing objects into piles based on their shared characteristics. This technique is often used for *unsupervised learning*, where the algorithm is not given labeled data. Instead, the algorithm must discover the patterns in the data on its own and group the data points accordingly.
While classification and clustering are distinct machine learning techniques, they share a common thread: their ability to extract patterns and insights from data. Both techniques seek to understand the data's structure, but they approach this task from different angles.

Given the challenges presented by the task of classifying malicious comments, the expected outcome of the project is to verify whether malignant comments truly contain specific patterns and features that can be automatically detected, or if their semantic and other linguistic elements, such as irony, plays a big part in biasing the algorithm's results. By leveraging both techniques on malignant comments detection, we will try to gain a comprehensive understanding of the online abuse landscape and eventually understand deeper differences and similarities of the two methods.

3 Data

The data was collected from Kaggle [1]. The data is composed by two datasets: a training set, which has approximately 159,000 samples and the test set which contains nearly 153,000 instances. All the data samples contain *8 fields* which include:

- Malignant: it is the Label column, which includes values 0 and 1, denoting if the comment is malignant or not.
- Highly Malignant: it denotes comments that are highly malignant and hurtful.
- Rude: it denotes comments that are very rude and offensive.
- Threat: it contains indication of the comments that are giving any threat to someone.
- Abuse: it is for comments that are abusive in nature.
- Loathe: it describes the comments which are hateful and loathing in nature.
- ID: it includes unique Ids associated with each comment text given.
- Comment text: this column contains the comments extracted from various social media platforms.

The label can be either 0 or 1, where denotes 0 a *no* while 1 denotes a *yes*. There are various comments which have multiple labels. The first attribute is a unique ID associated with each comment.

3.1 Data Exploration

The histogram shown in the figure below serves as a visual representation of the various factors contributing to comment toxicity across the six specified columns. Each single plot corresponds to a distinct aspect, that is a columns of the dataset. The separation between toxic and non-toxic comments is graphically emphasized by the color choice, where *green* indicates non-malignant comments and *red* represents toxic comment. It is clear that the dataset is unbalanced from this point of view.

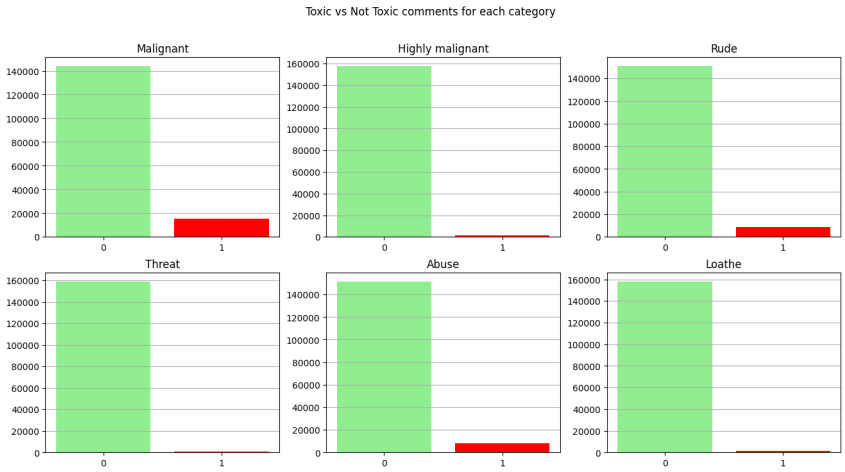


Figure 1: Toxic vs Not Toxic comments for each category

Overall, the graph shows that the vast majority of comments are not toxic. However, there is a significant amount of abusive language on the platform, with malignant, rude, and abuse being the most common types.

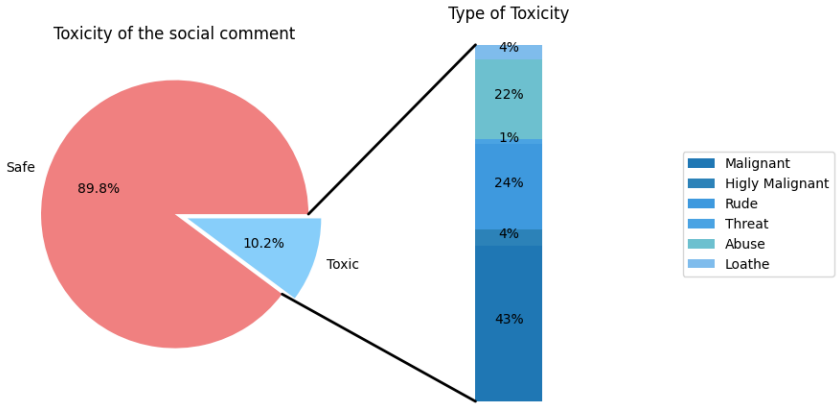


Figure 2: Toxicity of the social comment

The pie chart shows that the vast majority of social media comments are safe, at 89.8%. The other categories of toxicity have much lower percentages, 10.2%.

In this dataset, a substantial class imbalance is evident, with messages containing negative content notably being fewer in number. For representation purposes, at least one negative class is set to 1.

The presence of class imbalance poses a challenge, as the model might have a bias towards the majority class, potentially impacting its ability to effectively identify instances of the minority class. In scenarios where the class representing negative content is underrepresented, the model may exhibit a tendency to prioritize the majority class in its predictions. Therefore, special attention is required to ensure the model's proficiency in recognizing both positive and negative instances.

Moving forward, the correlation matrix provides insights into the associations between various categories of toxicity in the dataset. Notably, there are strong positive correlations between `rude` and `abuse` (0.74), as well as `Toxicity` and `rude` (0.70). The `highly_malignant` category shows moderate positive correlations with `rude` (0.403) and `abuse` (0.376). It is noteworthy that none of the variable pairs exhibit a negative correlation.

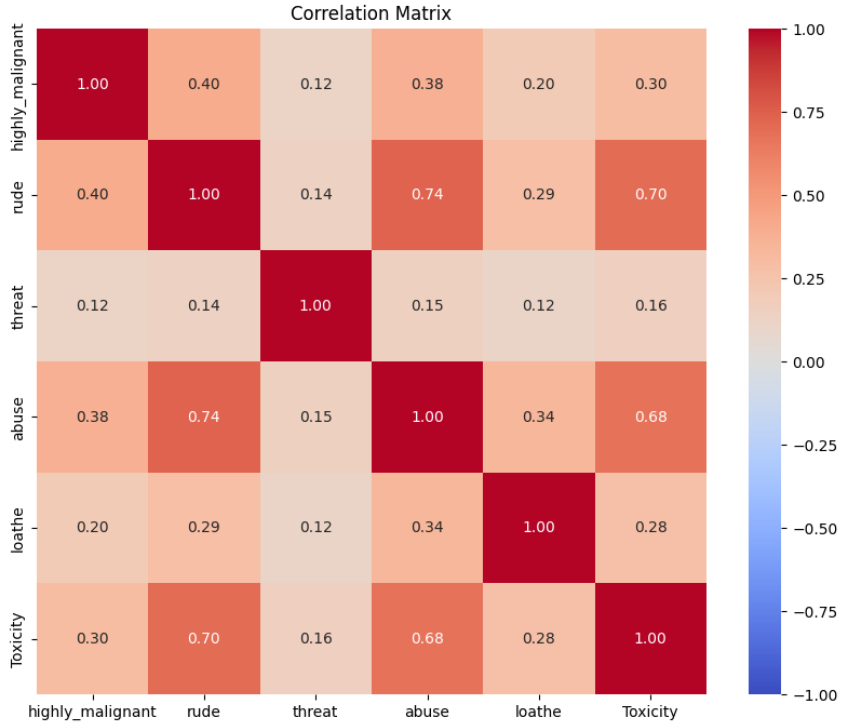


Figure 3: Correlation between the malignant attribute

From this point onward, to increase the number of negative comments, we consider any comment that has at least one of the attributes set to 1 as `malignant`.

4 Text Preprocessing

The aim of the text preprocessing phase is to clean and preprocess textual data, making it more suitable for natural language processing (NLP) tasks such as sentiment analysis or classification. In this case, the resulting preprocessed text is stored in a new column named `text_preprocessed` in the DataFrame.

1. **Remove URLs**: the regular expression pattern `https?://\S+www\.\S+` is used to identify and remove URLs from each comment in the `comment_text` column.

2. **Replace Newlines**: this phase replaces newline characters (\n) in the preprocessed text with a space, ensuring that the text remains continuous and coherent.

3. **Convert to Lowercase**: the text is converted to lowercase to ensure uniformity and eliminate case sensitivity in subsequent analyses.

4. **Strip Whitespace**: leading and trailing whitespaces are removed from each comment.
5. **Tokenization**: the NLTK library's function is applied to tokenize the text into individual words, creating a list of tokens for each comment.
6. **Remove Stopwords**: common English stopwords (e.g., 'the,' 'and') are removed from the tokenized text to focus on more meaningful words.
7. **Remove Special Characters**: any remaining special characters are removed from the tokenized and stopword-removed text.
8. **Remove Long Tokens**: tokens that are excessively long are filtered out, possibly to handle outliers or anomalies in the text.
9. **Lemmatization**: words are lemmatized to reduce them to their base forms, enhancing consistency and improving the efficacy of downstream analyses.
10. **Remove Empty Tokens**: any remaining empty tokens resulting from the preprocessing steps are handled and removed.

The plot that illustrates the frequency of text lengths before and after preprocessing.

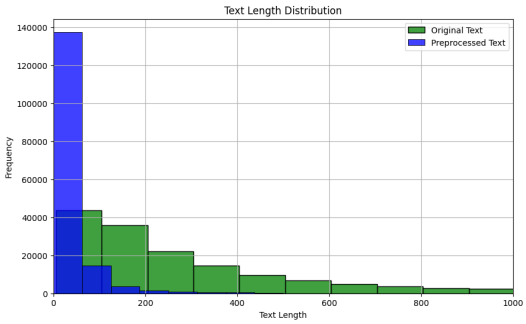


Figure 4: Text length (frequencies) before and after preprocessing

At first glance, it is possible to see that preprocessed comments are much shorter that the original ones. This is because preprocessing typically involves removing stop words, punctuation, and other unnecessary characters from the text. The decrease in text length after preprocessing suggests that the preprocessing steps were effective in removing unnecessary characters and features from the text. This is likely to lead to improved accuracy and improved interpretability of the data.

Preprocessing is also important because it can help to reduce the dimensionality of the data. Dimensionality refers to the number of features in the data. High-dimensional data can be difficult and computationally expensive to process.

The two word cluod plots below wants to highlight the different language used in malignant and non malignant comments.

As it it possible to observe, non-malignant comments usually do not contain any malicious word and these comments contribute to positive and constructive communication in

(a) Non-malignant comments

(b) Malignant comments

Figure 5: Comparison between the most used words in for each category.

various online platforms. On the other hand malignant comments contain words that exhibit harmful, offensive, or malicious intent, pointing out the importance of their removal from conversation and the Web in general.

5 Classification

In this section, the Text classification task will be illustrated. It has the goal of predicting to which of the predefined set of groups, or categories, a given textual data item belongs. Specifically in our case, a binary classification is performed, signifying a distinct categorization into one of two specific classes: **Malignant** and **Not Malignant**. The emphasis here lies on hard classification, meaning that each text item is unambiguously assigned to a particular category.

This binary classification endeavor is particularly pertinent for our specific text-based dataset. The aim is to train a machine learning model to discern intricate patterns within the textual data, enabling it to make precise predictions regarding the category of new, unseen text entries.

Our approach involves employing algorithms that can efficiently learn and distinguish features indicative of the two defined categories. The model, once trained, becomes a reliable tool for automated decision-making, offering a practical solution for tasks demanding a clear-cut classification of textual data.

5.1 Model Selection

In this phase, a thorough evaluation of different models is conducted to select the one demonstrating superior Recall for the final test. Various models will be explored to understand which one best fits the given data. The objective is to identify a model showcasing the best predictive capability on the validation dataset to ensure an accurate model during the testing phase. This approach aims to maximize the model's effectiveness in generalizing and adapting to the specific data characteristics of the problem at hand.

In the current phase, the use of four models is planned: **Logistic regression**, **Random forest**, **XGBoost** (xgb), and **Adaboost** (ada). This diverse model selection aims to cover various approaches and characteristics of each algorithm.

Regarding the issue of imbalanced datasets, the use of ensemble learning techniques, such as Random Forest, XGBoost, and Adaboost, is particularly advantageous. Ensemble algorithms are well-known for their ability to effectively handle imbalanced datasets.

For each of these models, a phase of searching for the best parameters is conducted using **grid search CV** (Cross-Validation). This process involves systematically exploring various combinations of hyperparameters through a predefined grid and evaluating the model's performance using a cross-validation procedure. The initial dataset has been divided into a training set and a validation set, encompassing approximately 80% and 20% of the total dataset, respectively.

Then a **TfidfVectorizer** is applied on the text data. Tfidf is a feature extraction method that transforms a collection of raw documents into a matrix of TF-IDF (Term Frequency-Inverse Document Frequency) features. It essentially converts a set of text documents to a matrix of token counts or TF-IDF features with the maximum number of features (unique words or terms) to be extracted and included in the resulting matrix set to 15,000. Now, for each model, the best parameters and the results obtained from training with these parameters will be illustrated.

5.1.1 Logistic Regression

Logistic regression is a machine learning algorithm used to model the probability that a binary dependent variable takes one of two possible values (0 or 1) based on a linear combination of independent variables.

The output of logistic regression is transformed through the logistic (or sigmoid) function, compressing the output to a range between 0 and 1, representing the probability. If the estimated probability exceeds a predetermined threshold (usually 0.5), the instance is assigned to class 1; otherwise, it is assigned to class 0. Its interpretability and the ability to handle non-linear relationships through appropriate transformations make it a valuable tool in predictive analysis.

The tables showcase the performance of the text classification model. The overall accuracy stands at 93.4%, with a confusion matrix highlighting the correct classification of 27,144 negative instances and 2,658 positive instances.

Parameters	**Value**
solver	liblinear
penalty	l2
max_iter	300
class_weight	balanced
C	10

Table 1: Best Parameters Logistic Regression

Set	Accuracy
Training	0.935
Validation	0.934

Table 2: Accuracy Logistic Regression

	Predicted 0	Predicted 1
Actual 0	27144	1590
Actual 1	523	2658

Table 3: Confusion Matrix Logistic Regression

Class	Precision	Recall	F1-Score	Support
0	0.98	0.94	0.96	28734
1	0.63	0.84	0.72	3181
Accuracy: -, -, 0.93, 31915				
Macro Avg: 0.80, 0.89, 0.84, 31915				
Weighted Avg: 0.95, 0.93, 0.94, 31915				

Table 4: Classification Report Logistic Regression

5.1.2 Random Forest

Random Forest is a machine learning algorithm that belongs to the ensemble methods category. Essentially, Random Forest creates an ensemble of decision trees during the training process and provides predictions based on the voting or averaging of individual tree predictions.

The process of creating a Random Forest involves the random selection of subsets of the training data and the construction of decision trees on these subsets. This random sampling technique, known as "bagging" (Bootstrap Aggregating), helps reduce overfitting and improve the model's generalization.

Each tree in the Random Forest operates on a random subset of features, contributing to the diversification of trees and making the model robust to various sources of variability in the data.

The training accuracy is 75.3%, while in testing, it is 74.4%. The confusion matrix reveals the correct classification of 21,064 negative instances and 2,696 positive instances.

Parameters	Value
n_estimators	100
min_samples_split	5
min_samples_leaf	4
max_features	log2
max_depth	20
class_weight	balanced
bootstrap	False

Table 5: Best Parameters Random Forest

Set	Accuracy
Training	0.753
Validation	0.744

Table 6: Accuracy Random Forest

	Predicted 0	Predicted 1
Actual 0	21064	7670
Actual 1	485	2696

Table 7: Confusion Matrix Random Forest

Class	Precision	Recall	F1-Score	Support
0	0.98	0.73	0.84	28734
1	0.26	0.85	0.40	3181
Accuracy: -, -, 0.74, 31915				
Macro Avg: 0.62, 0.79, 0.62, 31915				
Weighted Avg: 0.91, 0.74, 0.79, 31915				

Table 8: Classification Report Random Forest

5.1.3 XGBoost

XGBoost represents an advanced machine learning algorithm. XGBoost is an extension of the concept of gradient boosting, an approach that builds a predictive model by combining multiple simpler models. What makes XGBoost remarkable is its ability to enhance accuracy while maintaining a balance between model complexity and interpretability.

Among the distinctive features of XGBoost, its advanced regularization stands out, contributing to preventing overfitting through the use of L1 and L2 regularization terms.

The model exhibits high performance with a training accuracy of 95.7% and a commendable test accuracy of 95.3%. The confusion matrix indicates robust correct classifications, particularly in predicting negative instances (28,578 correct predictions out of 28,734). However, challenges are evident in predicting positive instances, with 1,852 true positives and 1,329 false negatives.

Parameters	Value
subsample	0.8
n_estimators	300
max_depth	4
learning_rate	0.2
colsample_bytree	0.8

Table 9: Best Parameters XGBoost

Set	Accuracy
Training	0.957
Validation	0.953

Table 10: Accuracy XGBoost

	Predicted 0	**Predicted 1**
Actual 0	28578	156
Actual 1	1329	1852

Table 11: Confusion Matrix XGBoost

Class	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.96	0.99	0.97	28734
1	0.92	0.58	0.71	3181
Accuracy: -, -, 0.95, 31915				
Macro Avg: 0.94, 0.79, 0.84, 31915				
Weighted Avg: 0.95, 0.95, 0.95, 31915				

Table 12: Classification Report XGBoost

5.1.4 ADAboost

Adaboost is a machine learning algorithm belonging to the ensemble methods category. The training process of Adaboost iterates through several phases. Initially, it assigns weights to each instance in the training set, giving more importance to examples misclassified in previous iterations. Subsequently, it trains a weak model, such as a decision tree, on the weighted training set.

During subsequent iterations, Adaboost focuses on difficult examples by increasing their weights and trains a new weak model. This process continues until a predefined number of weak models have been trained or until the error reaches an acceptable level.

In the end, Adaboost combines the weak models by assigning weights based on their performance, creating a strong model. The final output of Adaboost is determined by the weighted combination of the predictions of the weak models.

The model demonstrates strong performance with a training accuracy of 95.1% and a test accuracy of 95.2%. The confusion matrix reveals precise predictions for negative instances (28,457 correct predictions out of 28,734) but indicates challenges in correctly classifying positive instances, highlighting a trade-off between precision and recall in the classification report.

Parameters	**Value**
n_estimators	100
learning_rate	0.2

Table 13: Best Parameters ADAboost

Set	**Accuracy**
Training	0.951
Validation	0.952

Table 14: Accuracy ADAboost

	Predicted 0	**Predicted 1**
Actual 0	28457	277
Actual 1	1270	1911

Table 15: Confusion Matrix ADAboost

Class	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.96	0.99	0.97	28734
1	0.87	0.60	0.71	3181
Accuracy: -, -, 0.95, 31915				
Macro Avg: 0.92, 0.80, 0.84, 31915				
Weighted Avg: 0.95, 0.95, 0.95, 31915				

Table 16: Classification Report ADAboost

5.2 Model comparison

To evaluate the best machine learning model, the ROC curve has been used. The ROC curve can be particularly useful in the presence of imbalanced datasets. It represents the trade-off between the sensitivity and specificity of the model across different classification thresholds. In the context of imbalanced datasets, where one class can be much more numerous than the other, it is important to assess the model's ability to correctly handle both positive and negative examples.

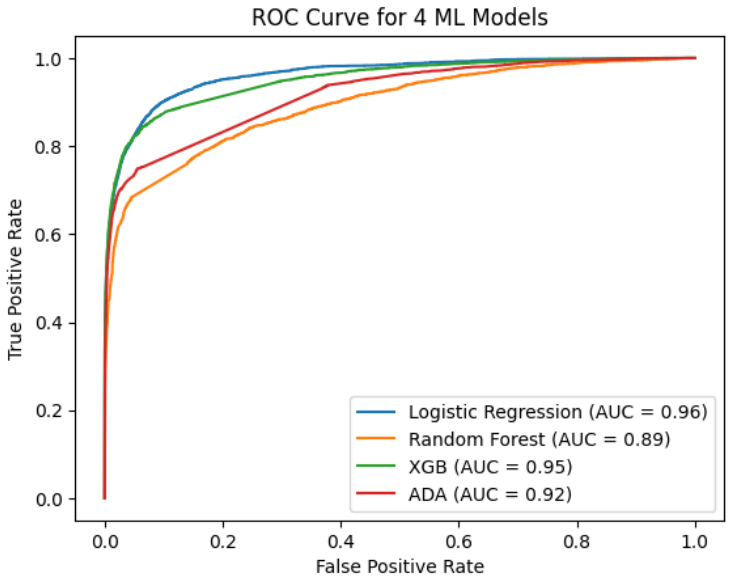


Figure 6: Roc Curve for the ML algorithm

The Area Under the Curve of the Receiver Operating Characteristic (AUC-ROC) is a key numerical indicator that summarizes the model's performance. A value closer to 1 suggests high accuracy in classification, while a value close to 0.5 indicates random performance. In this case the Logistic Regression is the model with value closer to 1, suggesting better performance.

5.3 ElectraSmall

At this point, the possibility of delving into a more advanced realm, that of deep learning, has been considered. The initial idea was to employ a BERT model for processing comments. Given its high computational cost, the implementation resolved to consider the use of ElectraSmall, that is a BERT-like model pre-trained as a discriminator of a generative adversarial network (GAN) [2], published by Kevin Clark, Minh-Thang Luong,Quoc V. Le and Christopher D. [3]

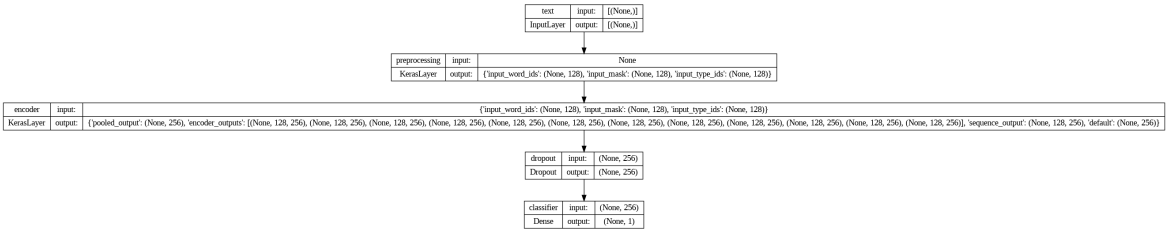


Figure 7: Electra small model

Below are described all the components of the model diplayed in the figure:

- **text input**: this layer defines the input for the model, which represents the text data to be processed. It takes in a string input of any shape.
- **preprocessing layer**: this layer utilizes the KerasLayer from TensorFlow Hub to preprocess the text input. It pointing to the preprocessing module for the BERT model on TensorFlow Hub. In this specific case, the preprocessing module is designed for the English-language BERT model [4]
- **encoder**: Load and utilize an encoder model pointing to the electra-Small model. The encoder processes the inputs and generates contextualized representations of the text.
- **dropout**: this layer applies a dropout regularization technique to the pooled output. Dropout randomly sets a fraction of input units to 0 during training, which helps prevent overfitting and improves generalization.
- **classifier**: this layer applies a fully connected dense layer with 1 unit (neurons). It performs a linear transformation on the input data, preparing it for classification. The 'sigmoid' activation is used to produce an output ranging between 0 and 1, which can be interpreted as the probability of belonging to one of the two classes.

At this point, the model has been compiled using:

- **Adam** as optimizer, which is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments [5]
- **Binary Crossentropy** as loss, which is commonly used in binary classification problems. This loss function evaluates the discrepancy between the probability distributions predicted by the model and the actual ones

- **Binary Accuracy** as metrics, that measures the model's accuracy in predicting binary classes compared to the true labels. Essentially, it represents the fraction of samples correctly classified relative to the total.

As the previously mentioned step, the dataset is split into training and validation sets, considering 80% and 20%, respectively. In this case, no additional preprocessing is performed as it is already incorporated within the layers of the neural network. During the training phase, three callbacks are taken into consideration:

- **ModelCheckpoint**: saves the model with the best validation performance during training.
- **EarlyStopping**: stops training if the validation binary accuracy does not improve for 4 consecutive epochs. This helps prevent overfitting.
- **ReduceLROnPlateau**: reduces the learning rate by a factor of 0.1 if the validation binary accuracy does not improve for 2 consecutive epochs. This can help the model converge more effectively by adjusting the learning rate dynamically.

5.4 Final results

As previously done, the model is evaluated considering the ROC curve.

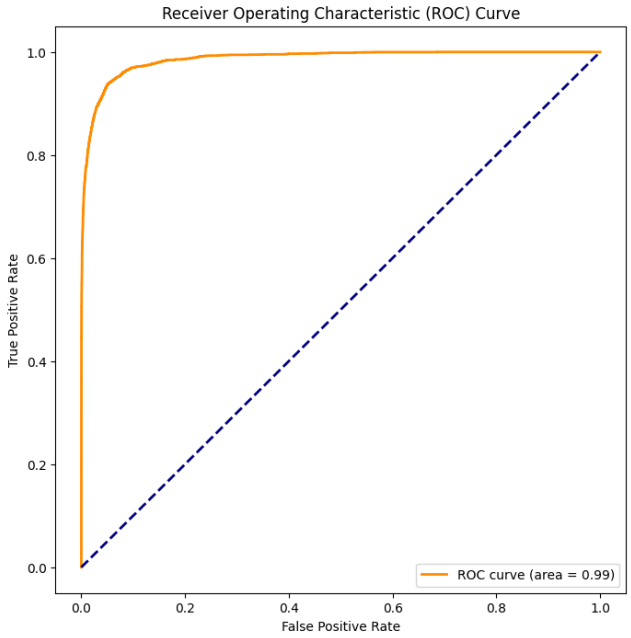


Figure 8: ROC Curve Electra small model

A value of 0.98 for the Area Under the Curve in a ROC curve is generally indicative of a highly accurate model. AUC measures the area under the ROC curve, representing the ability of a classification model to effectively discriminate between classes.

In this case, a value of 0.98 suggests that the model has very high performance, with high sensitivity and specificity in classification. The closer the AUC is to 1, the better the model's

ability to separate the target classes. Considering that among those previously proposed, it has the highest value, this model will be used for predictions on the test set.

At this point, leveraging the Electra Small model, predictions are made on the test set. This phase is crucial as it assesses how well the trained model generalizes to unseen data. By evaluating the model on the test set, which it has not encountered during training to avoid overfitting, we gain insights into its performance on real-world, previously unseen examples. The test set, containing 153,164 comments, serves as a robust benchmark for the model's effectiveness and generalization ability.

The focus here is on ensuring that the model's performance extends beyond the training data and is indicative of its real-world utility.

(a) wordcloud on test set

(b) Malignant comments

Figure 9: Comparison between the most used words in for each category.

By comparing the previously created word cloud plots with the current one generated from the test set, it becomes evident that the majority of words are consistent. This observation, combined with the fact that a significant portion of these words are explicit or offensive, further validates the accuracy of the model in identifying comments unfit for publication.

This alignment between the word distributions in the training and test sets underscores the robustness of the model's learning. The model demonstrates a commendable ability to generalize its understanding of inappropriate language across different datasets. The prevalence of offensive words in the identified comments indicates that the model has successfully learned and captured the patterns associated with objectionable content.

6 Clustering

As introduced above, text clustering is a machine learning technique that involves grouping similar pieces of text together. It is an unsupervised learning algorithm, meaning that it does not require labeled data to train on. Instead, the algorithm learns to identify patterns in the data and group documents based on those patterns.
Here we will try to group malignant and non malignant comments in two separate groups, without providing the algorithm any ground truth, so that it is possible to see if this techniques are able to distinguish between these two classes autonomously.

6.1 word2vec model and Vectorization

The word2vec model is a powerful tool for text clustering because it allows the algorithm to capture the semantic relationships between words. This is important because it allows the algorithm to group documents together based on their overall meaning, rather than just on the words that they contain. This representation is also able to capture the context of words. Therefore, it allows the algorithm to better understand the meaning of words in different contexts.

After a first preprocessing phase, the model was applied on the train dataset and the resulting vocabulary contains 39882 words. The word2vec representation allows the computation of similarity (or distance) between words and the resulting visualization.

Term	**Semantic Similarity**
truth	0.6844
falsehood	0.6611
lying	0.6580
misinformation	0.6445
slander	0.6430
propaganda	0.6404
liar	0.6392
twisting	0.6360
outright	0.6015
propagandist	0.5925

Table 17: Example of semantics similarity to the word *lie*.

The word2vec representation was implemented since it should improve the performance of text clustering algorithms.

However, the vectors produced by word2vec are not directly compatible with many machine learning algorithms, which typically require numerical data in a specific format.

This is where **vectorization** techniques come into play. For this reason, Gensim's Word Embedding was implemented: it captures word relationships based on their co-occurrence in the training corpus, effectively capturing contextual information. After the transformation, we obtain a list of vectors, one for each document and on length 300 (features).

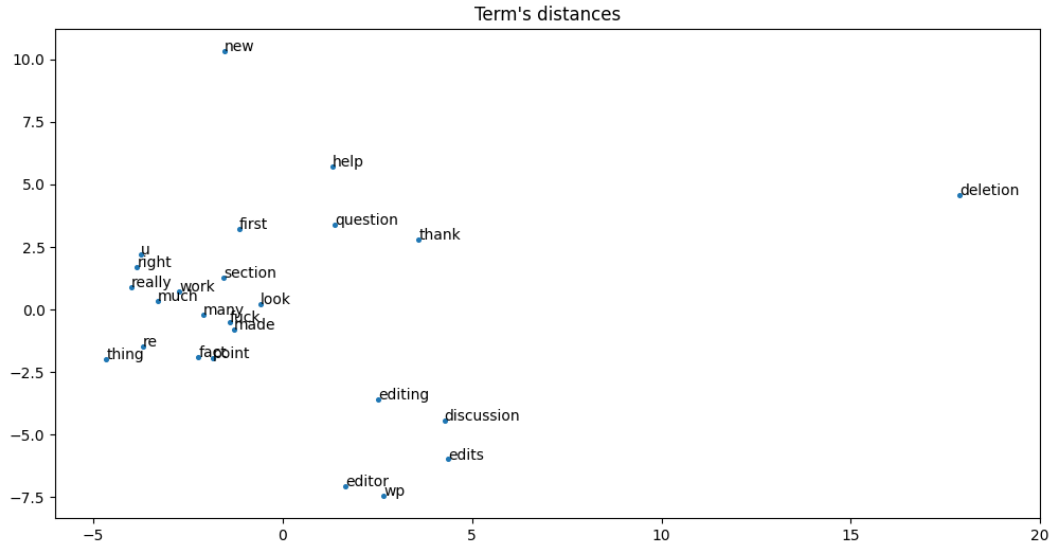


Figure 10: Visual example of semantic distance between words

6.2 Models applied

In the project, two different algorithm were implemented in order to compare their performances on the task: K-means and Fuzzy C-means. In particular, they are respectively a hard clustering algorithm and a soft clustering algotithm.

Hard clustering and soft clustering are two types of clustering algorithms that differ in how they assign data points to clusters.

Hard clustering algorithms assign each data point to exactly one cluster, meaning that a data point can only belong to one cluster, and the membership of a data point in a cluster is either 1 or 0. On the other hand, soft clustering algorithms assign each data point to a distribution over all clusters. Thus, a data point can belong to multiple clusters, and the membership of a data point in a cluster can be any value between 0 and 1.

6.2.1 K-Means algorithm

K-means clustering is a partitional clustering algorithm, meaning that it aims to partition the data into a predefined number of clusters (k) by iteratively assigning data points to the nearest cluster centroid and then updating the centroids to reflect the mean of the assigned points. This process is repeated until the clustering converges, meaning that the assignments of data points to clusters no longer change.

The number of clusters was set to two, indicating the two types of comments we are analysing. The model was trained on the training set and then tested on the test set. On the latter dataset, the algorithm returned a **Silhouette coefficient value of 0.118**, indicating that the clustering results are not really strong. This means that the clusters are not well-separated and that there is a high degree of overlap between them. This could make it

difficult to interpret the text, given their complex semantic.

However, by analysing the results we can see that model is still able to capture some interesting features from comments. First of all, the resulting cluster's sizes are alienated with the original data. In particular, from the dataset we know that the two classes are heavily unbalanced, since the non-malignant comments are present in a really much higher number with respect to malignant comments. The algorithm is able to catch this characteristic, even if the scale of the phenomena is underestimated.

Cluster	**Size**
0	54728
1	98436

Table 18: Cluster Sizes

Note: in contrast to the previous statement, here we will consider 1 to be *non-malignant* and 0 to be *malignant* here.

The most representative terms per cluster can provide valuable insights into the nature of malignant comments by helping to interpret the results of the clustering algorithm. This can be helpful in understanding the nature of the malignant comments and identifying the patterns that distinguish them from benign comments. The most representative terms can also provide clues about the topics, themes, and linguistic features that are associated with different types of malignant comments.

Below are reported the most representative terms for each cluster.

Cluster	**Most Representative Terms**
0	brag, fatass, confront, ure, mokele
1	article, anyway, necessary, particular, otherwise

Table 19: Most Representative Terms per Cluster

The words representing **Cluster 0** suggest that this cluster may contain comments that are aggressive, insulting, or abusive. These terms are often used to attack or demean others, and they can create a hostile and intimidating environment. The word *"ure"* may be a slang term for *"you"* or *"your"*, and its inclusion could further emphasize the accusatory or confrontational tone of the comments.

On the other hand, the words symbolizing **Cluster 1** suggest that this cluster may contain comments that are non-harmful and may contribute to the conversation in a meaningful way. These terms are often used in filler language, and their inclusion could indicate that the comments are trying to add something meaningful to the conversation. The word *"article*" could suggest that the comments are attempting to discuss about a topic. The words *"anyway"* and *"otherwise"* could be used to introduce new information or empathize a concept. The words *"necessary"* and *"particular"* could be used to make points.

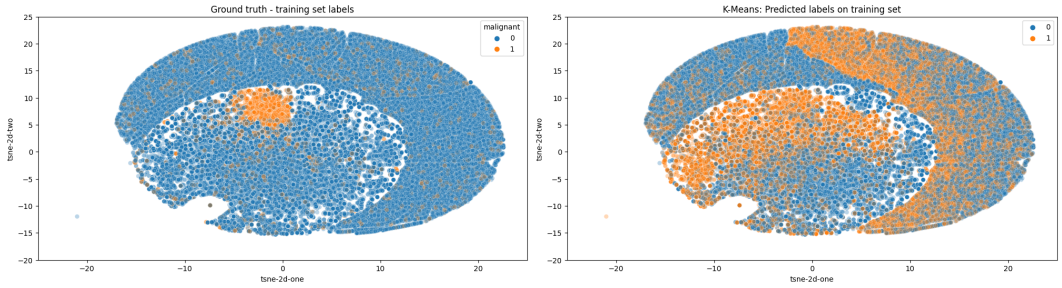


Figure 11: Comparison between the ground truth and clustering results on training set (K-Means)

The figure provide a visual representation of the result. For the clusters visualizations, t-SNE (t-Distributed Stochastic Neighbor Embedding) was employed. This approach is favored for dimensionality reduction and visualization due to its ability to effectively capture and showcase intricate patterns in high-dimensional data. Unlike linear methods such as PCA, t-SNE is non-linear, making it adept at preserving local structures and revealing complex relationships. It excels in visualizing clusters, helping identify distinct groups within the data. Given its flexibility, t-SNE was able to produce better plots.

The two plots show respectively the ground truth labeling of training data and the results of K-means clustering on training data. In the best case scenario, we would expect the labels from the clustering results to be distributed in a similar way to the ones from the original dataset. However, this situation did not occur.

The first thing to notice is that the K-means clustering results are not aligned with the ground truth labels, even on the training data. This is because K-means clustering is an unsupervised learning algorithm, and it does not have any prior knowledge of the true labels of the data. Instead, it clusters the data based on the similarity of the data points, as measured by some distance metric. We could infer that malignant and non-malignant comments are not so unlike, or at least their representations are not that contrasting. In fact, from the image we can see that they overlap.

This suggests that the K-means clustering algorithm may not be able to identify a subset of malignant comments with high accuracy and it may misclassify malignant comments as non-malignant, and vice versa.

The results of the application of the trained K-Means algorithm on the test set confirm what discussed above.

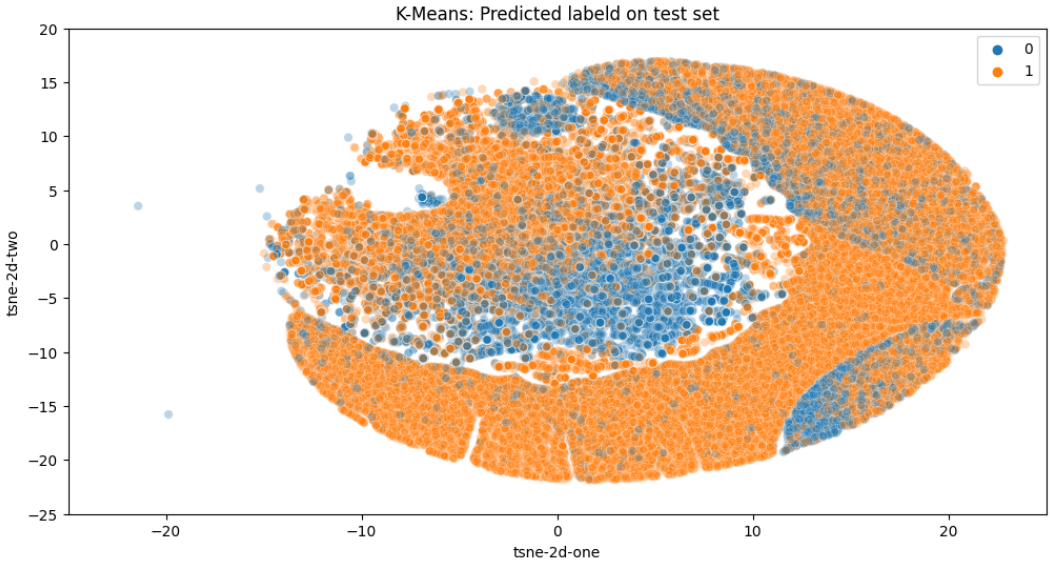


Figure 12: Predicted labels on test set with the K-means algorithm.

6.2.2 Fuzzy C-means algorithm (FCM)

Given the previous results, another algorithm was introduced in order to try to improve the performances of clustering.

C-means clustering is a generalization of k-means clustering that allows for the number of clusters to be learned from the data rather than being predefined. This is done by introducing a fuzzy membership parameter for each data point, which allows each data point to belong to multiple clusters with different degrees of membership. The algorithm then iteratively optimizes the cluster centroids and membership parameters to minimize a cost function that measures the overall dissimilarity between the data points and their assigned clusters.

Unlike traditional clustering algorithms that assign data points to a single cluster, FCM allows each data point to belong to multiple clusters with varying degrees of membership. This makes FCM better suited for capturing the nuances of real-world data, which often does not conform to strict, hard-edged boundaries.

In general, C-means clustering is more versatile than k-means clustering, as it can handle a wider range of data distributions and can be more robust to outliers. In order to maintain coherence, also this algorithm was provided with a given number of clusters (2) to compare the results with the K-Means algorithm.

For the implementation, the `scikit-fuzzy` library was exploited [6].

It is important to highlight that this algorithm required its own preprocessing steps. In particular, word2vec was not used. Instead, the preprocessed text used used for the classification was de-tokenized transformed into both a Bag-of-Words model and subsequently a TF-IDF matrix with a limited vocabulary. Afterwards, it employs Truncated SVD to perform dimensionality reduction on the TF-IDF matrix, resulting in a two-dimensional

representation. The normalization step enhances the stability of the reduced features. This approach was not implemented for the application of the K-Means algorithm in the final solution since it provided even worse performances.

To assess the clustering quality, the **Fuzzy Partition Coefficient (FPC)** was used. It provides insights into the degree of separation and distinctiveness of clusters in a fuzzy clustering solution.
The Fuzzy Partition Coefficient (FPC) is calculated using the formula:

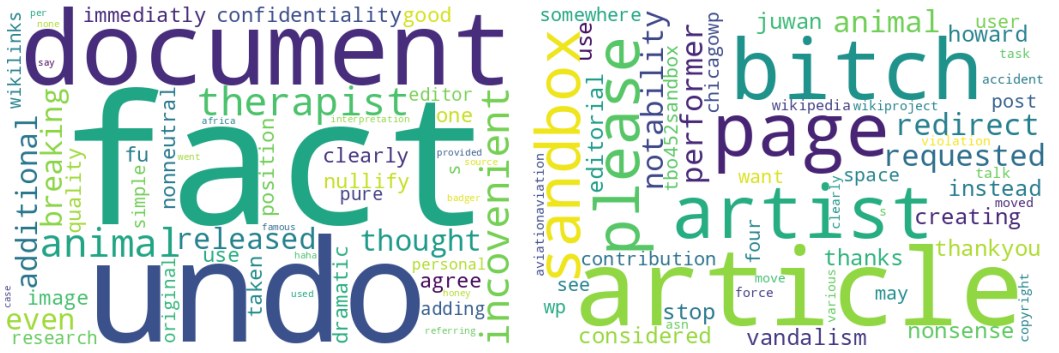
$$FPC = \frac{\sum_{i=1}^{N} \max_j(u_{ij})}{N}$$

where:

- N: number of data points
- u_{ij}: membership value of data point

It ranges between 0 and 1. A higher FPC suggests a clustering solution where data points distinctly align with their dominant clusters, indicating well-defined and separated groups. On the contrary, a lower FPC hints at a scenario where the boundaries between clusters are more diffuse, and data points exhibit a higher degree of ambiguity in their cluster assignments.
In this case, where the **FPC is approximately 0.82**, suggesting that the fuzzy clustering result is favorable.

Below are illustrated the most important words for each cluster.

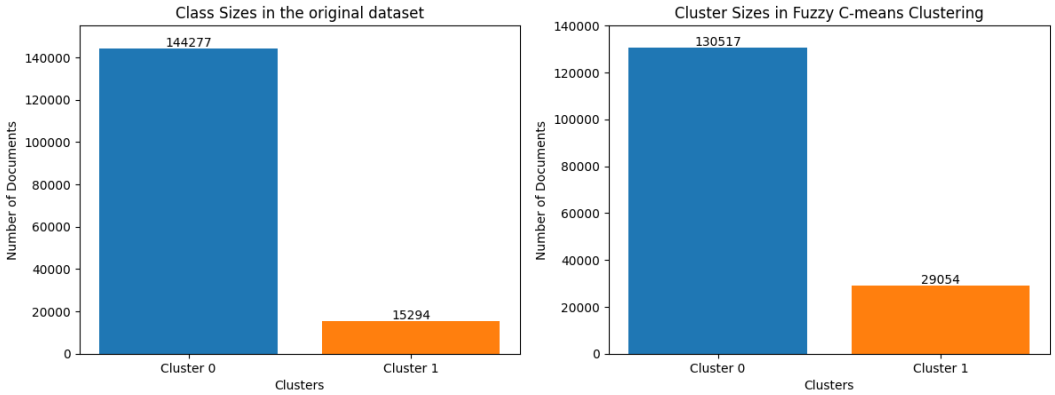


(a) Cluster 0 (b) Cluster 1

Figure 13: Comparison between the most important words in each cluster.

On the contrary of the words seen for the K-Means algorithm, here it is not possible to identify a net distinction between the words from the two clusters. All the words seem generic and they are not particularly offensive, nor indicate extremely polarized opinions.

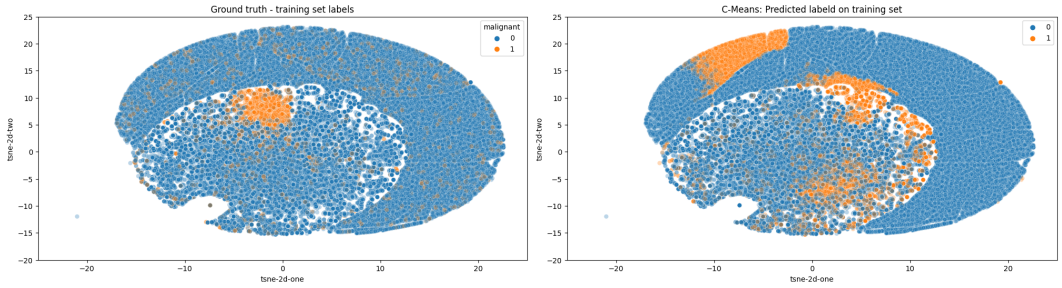
Also in this case, the algorithm is able to identify the fact that the dataset is unbalanced.



(a) Original labels (b) C-means predicted labels

Figure 14: Comparison between original classes and predicted classes.

The overlap between the sizes of the clusters suggests that there are some comments that are similar to both clusters and that the algorithm may *misclassify* some non-malignant comments as malignant, and vice versa. This is due mostly to the fact that natural language is complex and and not straightforward. We should recall that this approach aims to evaluate the algorithm's ability to independently distinguish between these two classes solely based on inherent patterns present in the data, and it is also possible that between these two classes there are not a lot of differences from this point of view.



(a) Original labels (b) C-means predicted labels

Figure 15: Comparison between the ground truth and clustering results on training set (C-Means)

Despite this, by observing the comparison between the ground truth and clustering results on training set, it is possible to see some similarities.

This suggest that the algorithm is able to identify the distinction between malignant and non-malignant comments, even is this division in not perfect. This result empathize the conclusions derived from the FPC value, which suggested a fairly level of distinction.

This can be observed also from the predictions on the test set.

The distribution of the data points in the two clusters suggests that there is a significant overlap between malignant and non-malignant comments. This means that it may be difficult to develop a machine learning model that can accurately identify all malignant comments without also misclassifying some non-malignant comments.

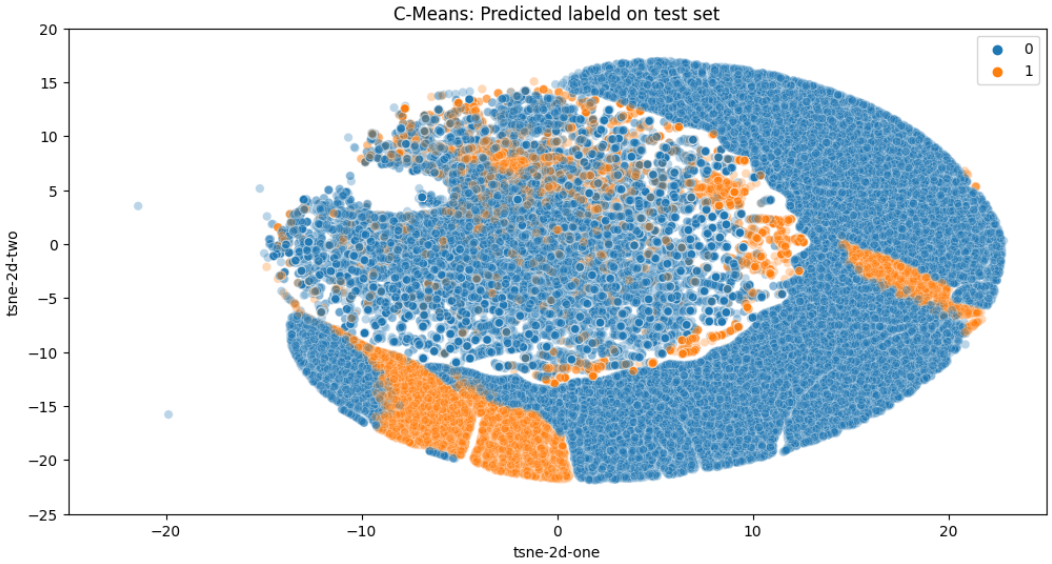


Figure 16: Predicted labels on test set with the C-means algorithm.

However, the fact that the two clusters are still somewhat distinct suggests that it may be possible to develop a machine learning model that can identify a subset of malignant comments with high accuracy.

The presence of outliers in the blue cluster suggests that the FCM clustering algorithm may misclassify some non-malignant comments as malignant. Unfortunately, this is a common limitation of clustering algorithms.

Overall, the results suggest that the Fuzzy C-means algorithms is a promising tool for this task.

7 Conclusions

As we navigate the ever-changing digital landscape, automated classification of malicious comments stands as a beacon of hope, offering a path towards a safer and more inclusive online environment. By harnessing the power of technology, we can reclaim the internet as a space for meaningful exchange, free from the scourge of malicious comments, and foster a truly global community of open dialogue and mutual respect.

Malignant comment identification using classification and clustering techniques demonstrates the effectiveness of machine learning in identifying and characterizing harmful online content. These conclusions highlight the potential of machine learning in addressing online toxicity and safeguarding online communities. As these techniques continue to evolve, their role in combatting online abuse and promoting safe and inclusive online spaces is expected to grow significantly.

Despite the challenges, the potential benefits of automated classification are undeniable. This will empower online platforms to better protect their users, promoting a more respectful and inclusive online community.

References

[1] Malignant Comment Classification

[2] Electra Small

[3] Kevin Clark and Minh-Thang Luong and Quoc V. Le and Christopher D. Manning: ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators, ICLR 2020.

[4] preprocessing layer

[5] Adam optimizer

[6] Fuzzy C-means - Module: cluster