CentraleSupélec

Ensemble Learning

Final Project
Cyberbullying classification

Team members and email:

Chiara Palma	chiara.palma@student-cs.fr
Shwetha Salimath	shwetha.salimath@student-cs.fr
Maria Isabel Vera Cabrera	maria-isabel.vera-cabrera@student-cs.fr
Lauren Yoshizuka	lauren.yoshizuka@student-cs.fr
Cheng Wan	cheng.wan@student-cs.fr

Git link:
https://github.com/chiaracodes/EL-assignment

March 10, 2022

CentraleSupélec

Contents

1 Introduction

2 Problem Statement

The objective of this project is to use ensemble methods based on common machine learning algorithms to best predict the target variable **cyberbullying_type**. This involves both multi-class and binary classification models. The multi-class model will classify the tweets based on the classes of cyberbullying type; the binary model will determine whether or not the tweet contains "abusive" language. In addition to creating an ensemble method, we will compare the various models and discuss their performance.

3 Preprocessing

The data set contains 47000 tweets, which have been labelled based to their context with respect to the following cyberbullying classes:

- Age
- Ethnicity
- Gender
- Religion
- Other cyberbullying type
- Not cyberbullying

The data is well-balanced as there are approximately 8000 tweets per class. This dataset is sufficiently large enough to achieve good prediction results. The preprocessing phase is a vital step in the machine learning pipeline as it ensures the data is ready for the model to learn from.

3.1 Data cleaning: Duplicates

It is necessary to clean the tweets to prepare them for the text analysis for classification. We started by exploring the data and the first thing we found was that there were duplicates. There were 47,692 rows initially but after we deleted duplicates, there were 47,656 instances. However, when we looked at the tweets, we found they were still not unique, meaning that there were tweets classified as different cyberbullying types. Given that we had no further information to determine the true class, we kept only the first cyberbullying type per tweet.

After this, we ended up with 46,017 tweets. We checked the distribution of the number of tweets per type and we observed that the class that was most affected after the removal of duplicates was "other cyberbullying" which was constantly repeated in the data. At the end, this class had around 6,200 tweets while the other classes had around 7,900 tweets each.

3.2 Text transformations

We applied several transformations on the tweets. We started by removing non-textual characters like punctuation, symbols, emojis, links and user tags. Then we converted all the text to lowercase and then eliminated stop-words. By removing stop-words we condense the dataset; there are a lot of unnecessary words that do not add value to the training of the model. Next, we applied word tokenization to split text strings into individual words for further NLP processing. After tokenization, stemming and lemmatization were applied to reduce the redundancy of words while maintaining their meaning. Stemming strips the prefix and suffix off words where applicable, and lemmatization ensures the root word is one that is found in the English language after it has been stemmed.

After this preprocessing steps, we found that 271 cleaned tweets were now empty, because they only contained special characters filtered at the beginning. We removed those examples as they would only introduce noise to the model. Even though some preprocessed tweets were now repeated, we decided to keep them as they would help the model by reinforcing the key words present in every type of cyberbullying.

3.3 Visual analysis

We used some visual techniques to highlight the key words in each type of cyberbullying. As a first tool, we used word clouds which represent the frequency of a word by their size and color in an image (Figure 1). This helps focus attention on the most representative words and get an idea on the type of content in each kind of tweet.

Although word clouds help us identify the most important words, they do not provide much information about the actual number of times they appear on tweets. For this, we build bar plots to visualize the frequency if the top 10 words in each cyberbullying type (Figure 2). In this graphs, we can observe not only which are the most common words but also by how much and how they differ in recurrence among them. We also built pivot tables to know the exact numbers associated with each of them and we detected some words such as "people" or "like" that appear in the top of several types of cyberbullying.

4 Feature Engineering

First, we assigned a number from 0 to 9 to each cyberbullying type, in case any of the classification models need numerical encoding in the target variable.

We then extracted some features that we considered interesting analyzing like the number of words, letters and vowels in each tweet. We calculate the median value for each of those 3 features and plotted the results in Figure 3.

We can see that religion and age tweets consistently have a higher frequency in those features, while other or no cyberbullying have the lowest amount.

4.1 TF-IDF matrix

We started by splitting the data in two data sets: 75% for training and the remaining 25% for testing.

We initialized the TF-IDF vectorizer with parameters min_df=0.002% and max_df=60%. Those parameters represent the minimum and maximum threshold of the proportion of documents a term has to appear in to be considered when building the vocabulary.

We fitted the vectorizer on the training data to create a sparse matrix consisting of the TF-IDF score for tokens from the tweets. Afterwards, we used the learned parameters to apply the transformation to the test data set. The output from this process on each data set is a list of tuples that represents the sentence number and the feature (word) number. For each of them, we assign a TF-IDF value that weights words based on relevance.

5 Multi-class Modelling

5.0.1 Tree-based models and Boosting methods

We tried different methods that we learned in the Ensemble Learning course. We included the most basic model we reviewed which is a decision tree, we added the ensemble method of Random Forests and finally we reviewed some boosting algorithms such as Gradient Boosting, Extreme Gradient Boosting, Adaptive Boosting and Light Gradient Boosting Machine. We trained all of them without specifying any hyper parameters beyond the default ones to get a baseline. We used them to predict on the test set and we calculated their accuracy score which is reported in Figure 4. We see that the model with the best performance is Random Forest, hence we decided to focus on this method.

5.0.2 Random Forest

For the purpose of optimizing this model's results, we used hyperparameter tuning, through cross-validated grid-search over a parameter grid. We tried different values for the number of estimators or trees and also for the percentage of features to consider when looking for the best split. We evaluated those different combinations of parameters in a 4-fold cross validation configuration to get more reliable results. In Figure 5 we can observe the impact in accuracy of each configuration, concluding that the best parameters were 64 trees and 20% max features to include.

We fitted the model again using the best hyperparameters found and we got the results shown in Figure 6 and 7. We observed that not only we had a good overall result, but also when we looked at the performance in each class, all of them exhibited very good metrics. The ones with the lowest values are cyberbullying and religion which had an F1 score of around 0.6 but the other classes had a value around 0.9, even some were very close to 1. The overall accuracy, precission, recall and F1 score is 85% which are outstanding results, considering we are tackling a multiclassification problem.

5.0.3 SVM

Support Vector Machine (SVM) is also applied to train and predict the dataset. SVM is a supervised learning model for assessing data in classification and regression analysis with associated learning algorithms. We use of ensemble methods in conjunction with SVM. Each SVM is trained independently using a bootstrapping method, and a final prediction of the ensemble is constructed by aggregating the predictions of the individuals.

SVM takes 3 hyperparameters as input, thus a thorough and expensive hyperparameter tuning is needed to fully leverage the edges of SVM. Grid search with 5-fold cross- validation was used to find the optimal hyperparameters. The best combination of hyperparameters we found is C: 10, gamma: scale, kernel: rbf. The highest accuracy score on test set that we got from SVM with ensemble technique is 0.849283 lower than SVM without ensemble. SVM used full data in the training process leads to more accurate models. Since bootstrap uses only part of this data to train individual model, there could be a high bias between the accuracy of each SVM. Moreover, there is an obvious drawback of SVM: it took more than one hour to fit the model and predict the result. This may be because the large scale of data for SVM in this case. The results are consolidated in Table 1.

5.0.4 k-Nearest Neighbors Bagging

We also tested a bootstrap aggregating (bagging) ensemble method using a k-Nearest Neighbors (KNN) weak learner to perform the classification task. We saw in class that tree-based models are greatly improved when a bagging method is applied, since the bias and variance are reduced thanks to the decorrelation of trees resulting from bootstrapped sampling. Hence, we had the intuition to try another simple model like the KNN model. We were willing to sacrifice on training time with this weak learner, in hopes to achieve decent results thanks to its stability under changes to training data. By adjusting the default number of estimators from 10 to 50, we augmented the accuracy score of the KNN Bagging classifier from 0.625 to 0.696. Additionally, by including the out-of-bootstap (OOB) error in the bagging model, we hoped to improve the error estimate without much additional cost, since it is computed easily and allows the model to be tested during training.

6 Binary class Modelling

After having performed multiclassificatio to detect which type of cyberbullying a tweet would correspond to, we moved towards binary classification to flag potentially harmful tweets. This time the target variable to predict would be whether the tweet can be classified as cyberbullying or not.

6.0.1 Tree-based models and Boosting methods

We again tried decision tree, random forest and several boosting methods, obtaining as a result that Random forest was the best model with an accuracy of 0.88 on the test set (Figure 8).

6.0.2 Random Forest

This time we decided to set a higher percentage for training so that the model can learn from more data and possibly perform better. Hence, we used a 80/20 split between the training and testing data set.

Having learnt from the hyperparameter tuning results in the multiclassification part, we decided to focus our parameter grid on the direction of values with highest potential. This time, we searched for higher number of estimators and we looked closer at lower proportions of max features to include. We obtained that the best accuracy in 4-folds cross validation was obtained with 64 trees and 10% max features considered (Figure 9).

After training the model with these best parameters and evaluating them on the test set we got an accuracy of 0.88. The F1 score is 0.93 for no cyberbullying and 0.6 for cyberbullying, for a weighted average of 0.87. The precision and recall of the model are 0.87 and 0.88 respectively. We observe this binary classification task obtained a better performance than the multiclassification problem even when we used the same framework. This is expected because the model only has to focus on separating two classes instead of 6 and as it has more examples for each of them, it can learn and generalize better.

7 Appendix: Figures and tables

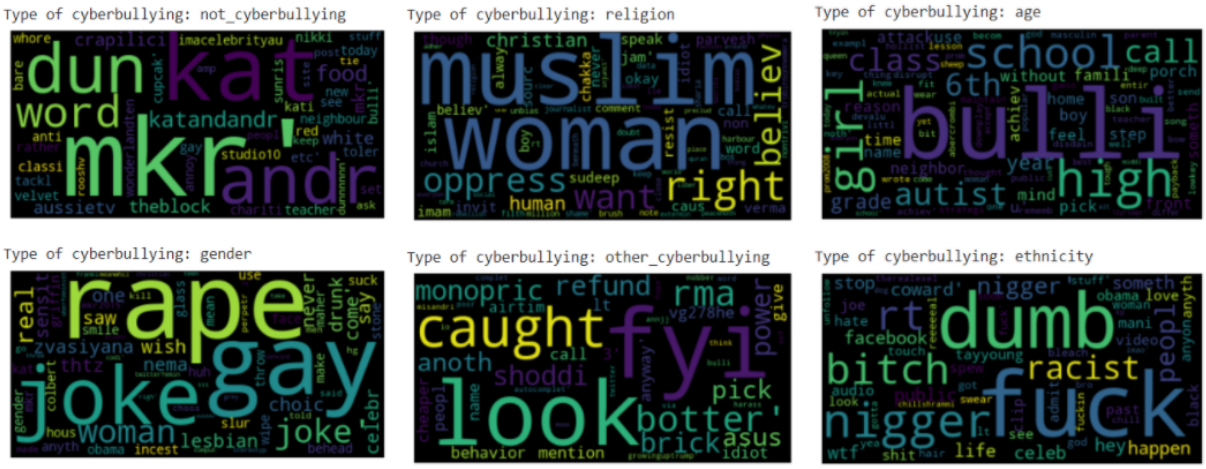

Figure 1: Word cloud per cyberbullying type

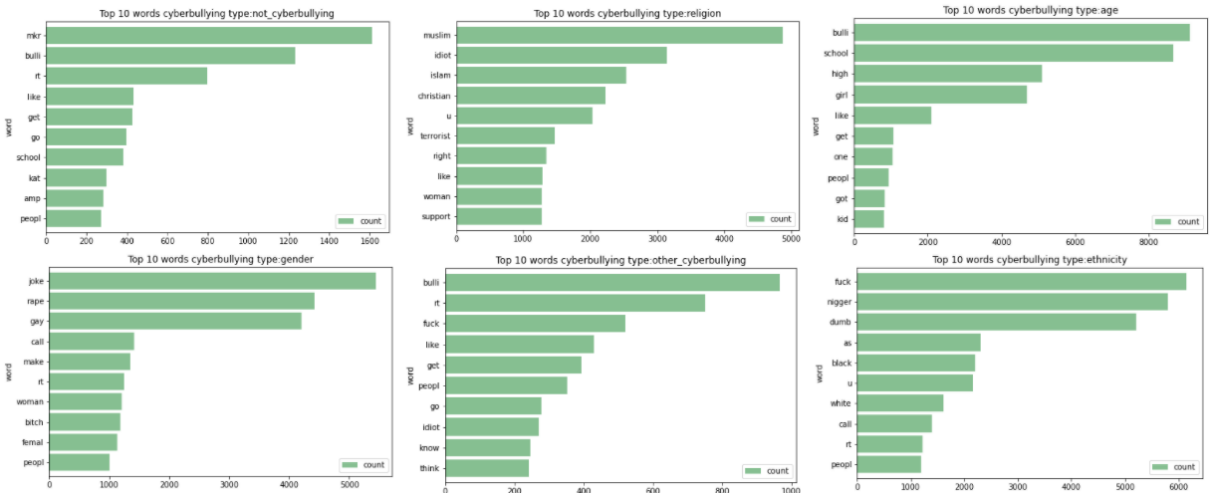

Figure 2: Word frequency plots per cyberbullying type

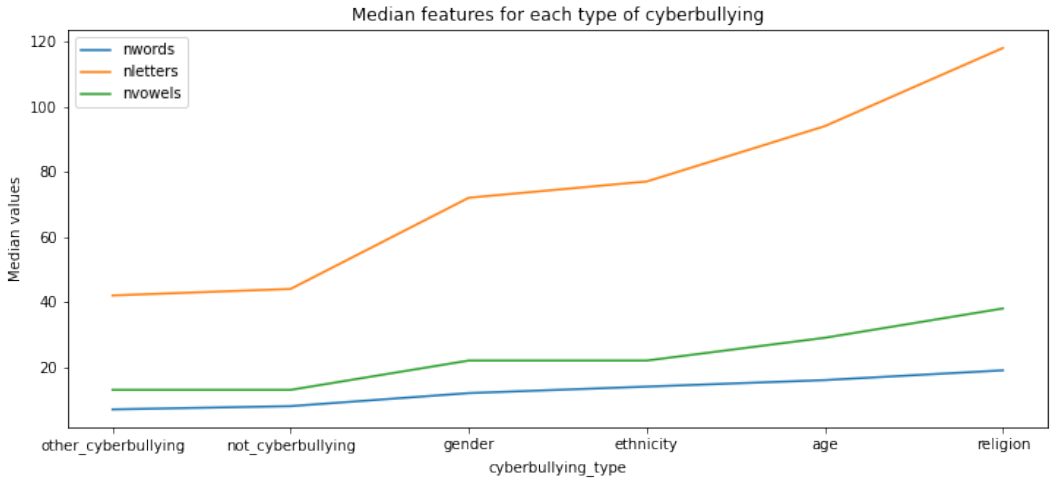

Figure 3: Median number of words, letters and vowels per cyberbullying type

Models	Accuracy Scores
Random Forest	0.849544
LightGBM	0.846675
Gradient Boosting	0.839983
Decision Tree	0.827553
XGBoost	0.827466
AdaBoost	0.796262

Figure 4: Model comparison based on accuracy in the test set

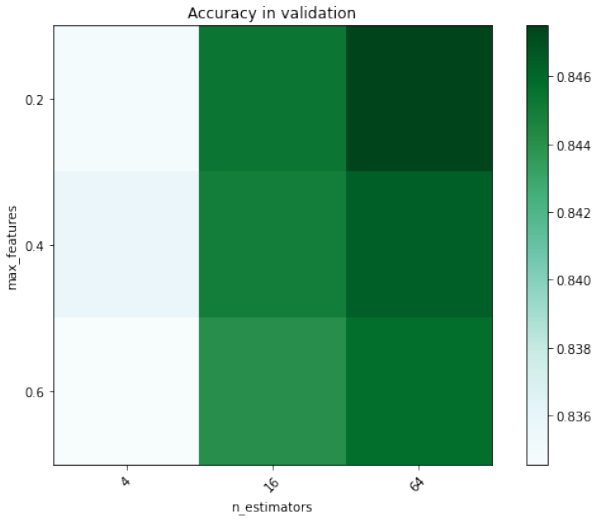

Figure 5: Hyperparameter tuning. Accuracy in cross validation

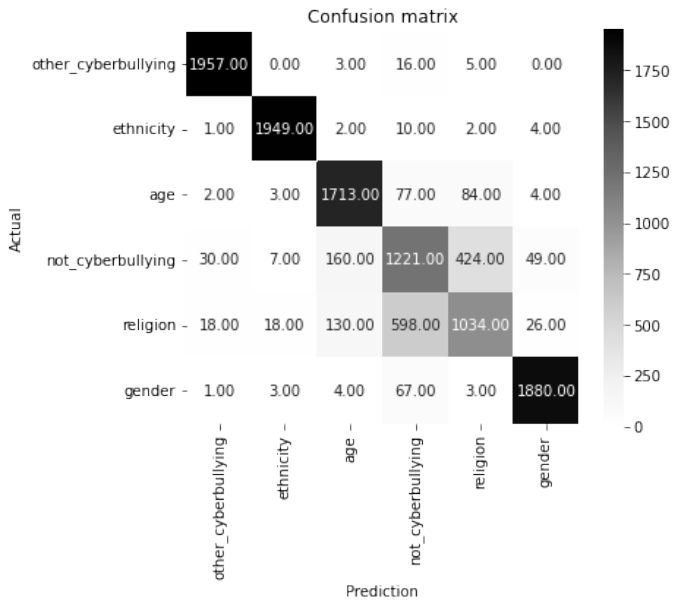

Figure 6: Hyperparameter tuning. Accuracy in cross validation

	precision	recall	f1-score	support
other_cyberbullying	0.98	0.98	0.98	2009
ethnicity	0.99	0.99	0.99	1980
age	0.91	0.85	0.88	2012
not_cyberbullying	0.65	0.61	0.63	1989
religion	0.58	0.66	0.62	1552
gender	0.95	0.97	0.96	1963
accuracy			0.85	11505
macro avg	0.84	0.84	0.84	11505
weighted avg	0.85	0.85	0.85	11505

Figure 7: Results report for the Random forest model

Table 1: Evaluation Metrics - SVM

	Accuracy on train	Accuracy on test	F1-score	Precision	Recall
SVM	0.975458	0.849283	0.851408	0.854557	0.849283

	Models	Accuracy Scores
1	Random Forest	0.879726
5	LightGBM	0.872881
0	Decision Tree	0.861691
2	Gradient Boosting	0.860604
4	AdaBoost	0.855280
3	XGBoost	0.852130

Figure 8: Binary model comparison based on accuracy in the test set

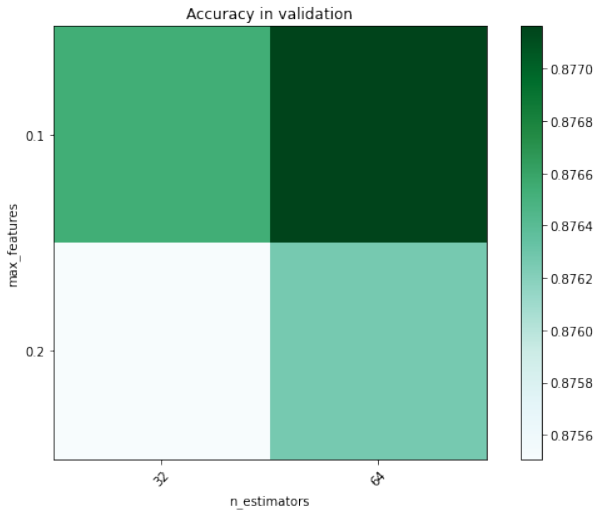

Figure 9: Hyperparameter tuning for binary classification. Accuracy in cross validation

	precision	recall	f1-score	support
not_cyberbullying	0.90	0.96	0.93	7616
cyberbullying	0.71	0.52	0.60	1588
accuracy			0.88	9204
macro avg	0.81	0.74	0.76	9204
weighted avg	0.87	0.88	0.87	9204

Figure 10: Results report for the binary Random forest model